Separated-Yet-Dense Random Point Clouds for Meshing and More

Scott A. Mitchell

www.cs.sandia.gov/~samitch

(or Google Mitchell Sandia)

2012 CSRI Summer Seminar Series

30 July 2012

3-4pm CSRI/90

Sandia National Laboratories

Abstract

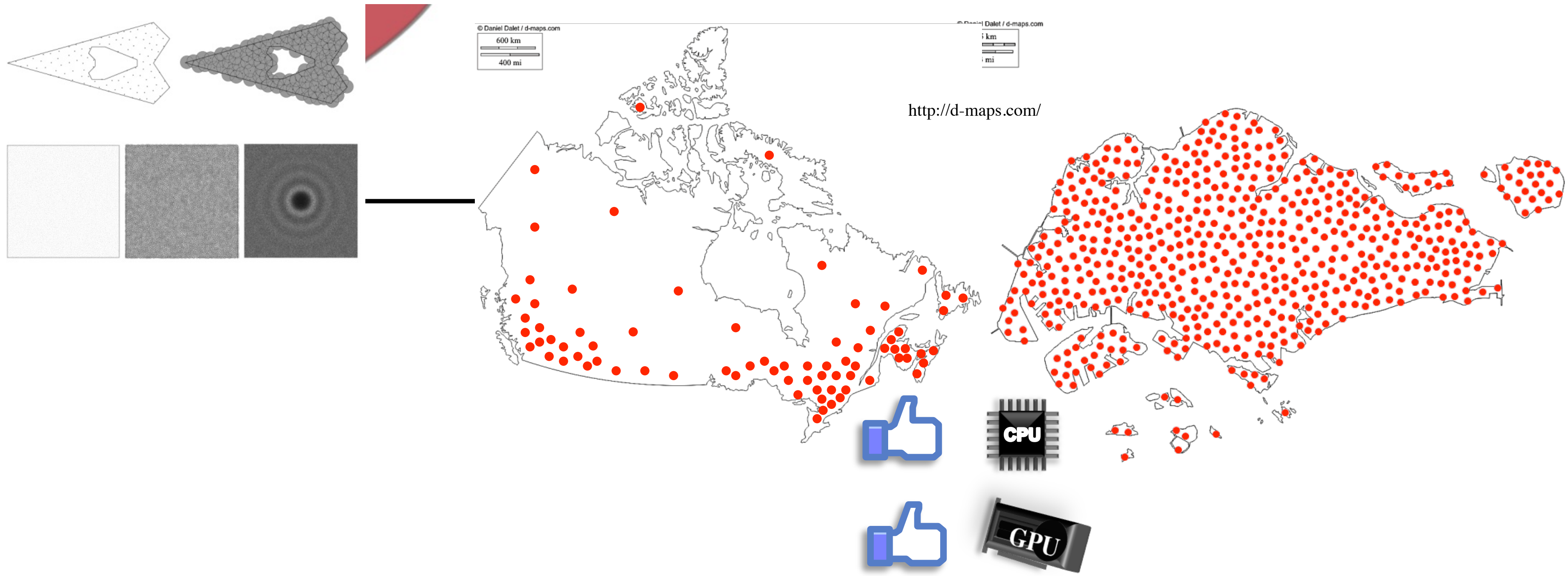
Dense-Yet-Separated Random Point Clouds for Meshing and More

Computational geometry is interesting to me because it combines both discrete and continuous objects, and both math and algorithms. I also like it because I can draw pictures to understand what I'm doing. Specifically I'll talk about the work we've done over the past couple of years on point clouds with random positions. We made up the term separated-yet-dense to describe sets of sample points such that no two points of the set are too close to one another, but any other point of the domain is close to some sample point. Computer Graphics has been obsessed with a particular way of generating these kind of point clouds, by selecting points sequentially and spatially uniformly at random. This way is important because it avoids visual artifacts in texture synthesis. Computational Geometry has been obsessed with a different way of generating these kinds of point clouds, by selecting them sequentially and deterministically, by selecting the domain point that is furthest away from the point cloud so far. Nearby points are attached together to generate a finite element mesh. The advantage of this approach is it is faster, and is easier to analyze. We've been coming up with algorithms that combine features of both approaches. Some have theory guarantees, and some are simpler and work better in practice. We have both computer graphics and mesh generation applications, and we've even started using random lines to efficiently solve some uncertainty quantification problems.

Sandia National Laboratories

Outline

- What is Maximal Poisson Disk Sampling MPS?
  - Graphics stippling and texture synthesis use
- Polygonal approximation algorithm (paper1)
  - Something provable
- Eurographics algorithm (paper2)
  - Simpler, better in practice, scales to high dim
- Define mesh, Delaunay triangulation, Voronoi diagram
- MPS for triangle meshing (paper3)
- MPS for dual Voronoi meshing (paper4)
- Variable radius, space and time
- Darts, QMU, … won’t get to



Efficient Maximal Poisson-Disk Sampling

Mohamed S. Ebeida, Anjul Patney, Scott A. Mitchell, Andrew A. Davidson, Patrick M. Knupp, John D. Owens

Sandia National Laboratories, University of California, Davis

Scott - presenter
SIGGRAPH2011

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Maximal Poisson-Disk Sampling

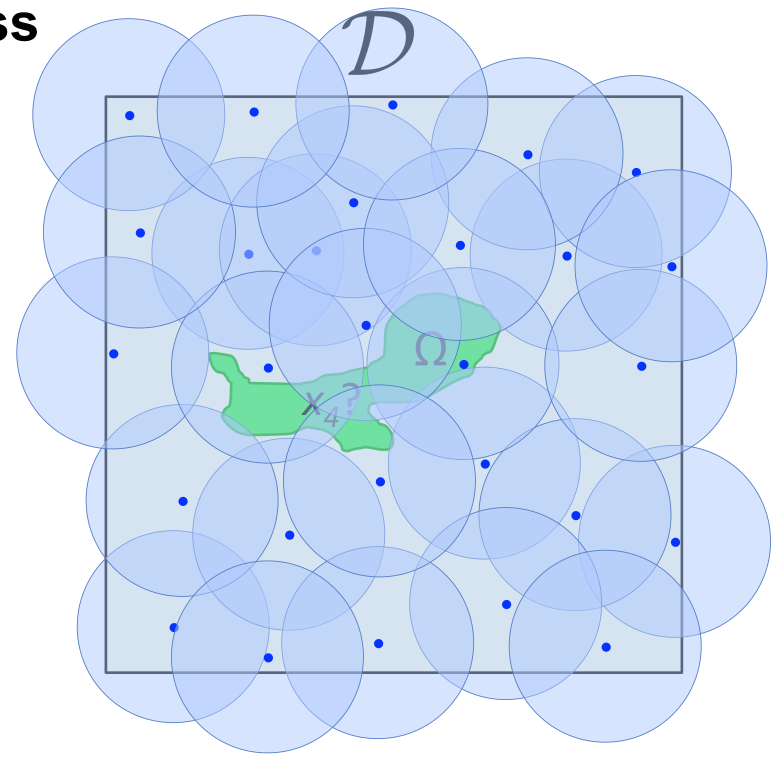
- **What is MPS?**
  - **Dart-throwing**
  - **Insert random points into a domain, build set X**
    - **With the "Poisson" process**

Empty disk: $\forall x_i, x_j \in X, x_i \neq x_j : ||x_i - x_j|| \geq r$

Bias-free: $\forall x_i \in X, \forall \Omega \subset \mathcal{D}_{i-1}:$

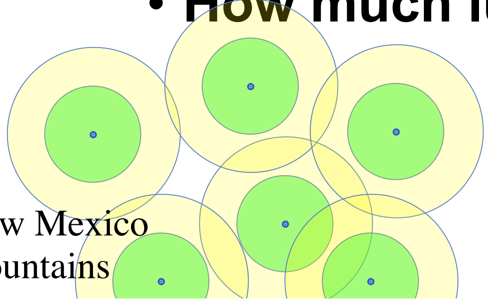
$$P(x_i \in \Omega) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$$

Maximal: $\forall x \in \mathcal{D}, \exists x_i \in X : ||x - x_i|| < r$

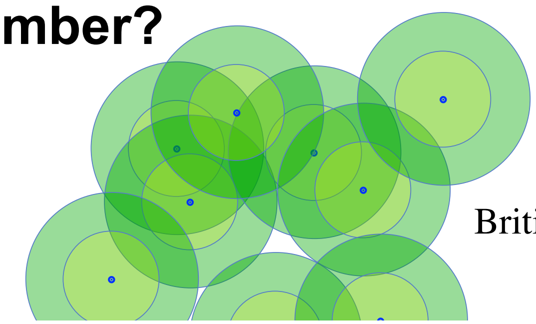


MPS a.k.a.

- Statistical processes
  - Hard-core Strauss disc processes
    - Non-overlap: inhibition distance r_1
    - cover domain: disc radius r_2
- Nature
  - Trees in a forest
    - Variable disk diameter = tree size
    - Points are tree trunks
    - Disks are tree leaves or roots
  - Given satellite pictures (non-maximal)
    - How many trees are there?
    - How much lumber?

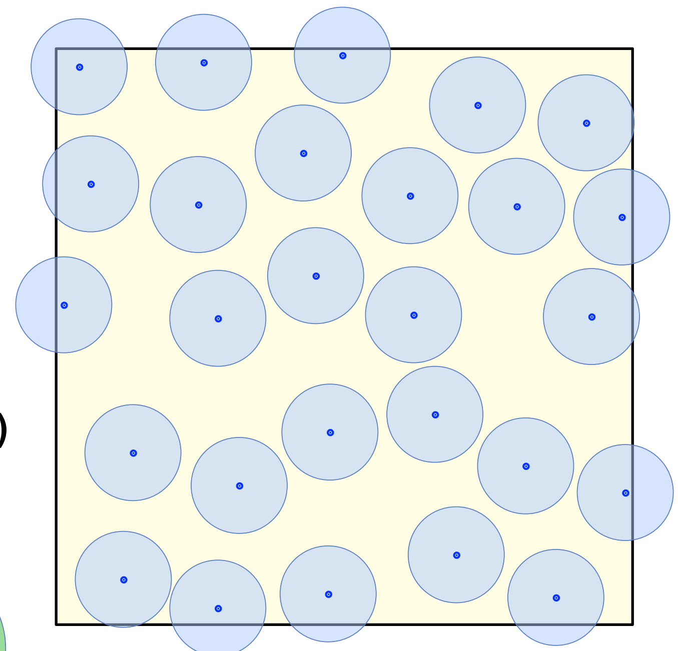


New Mexico mountains



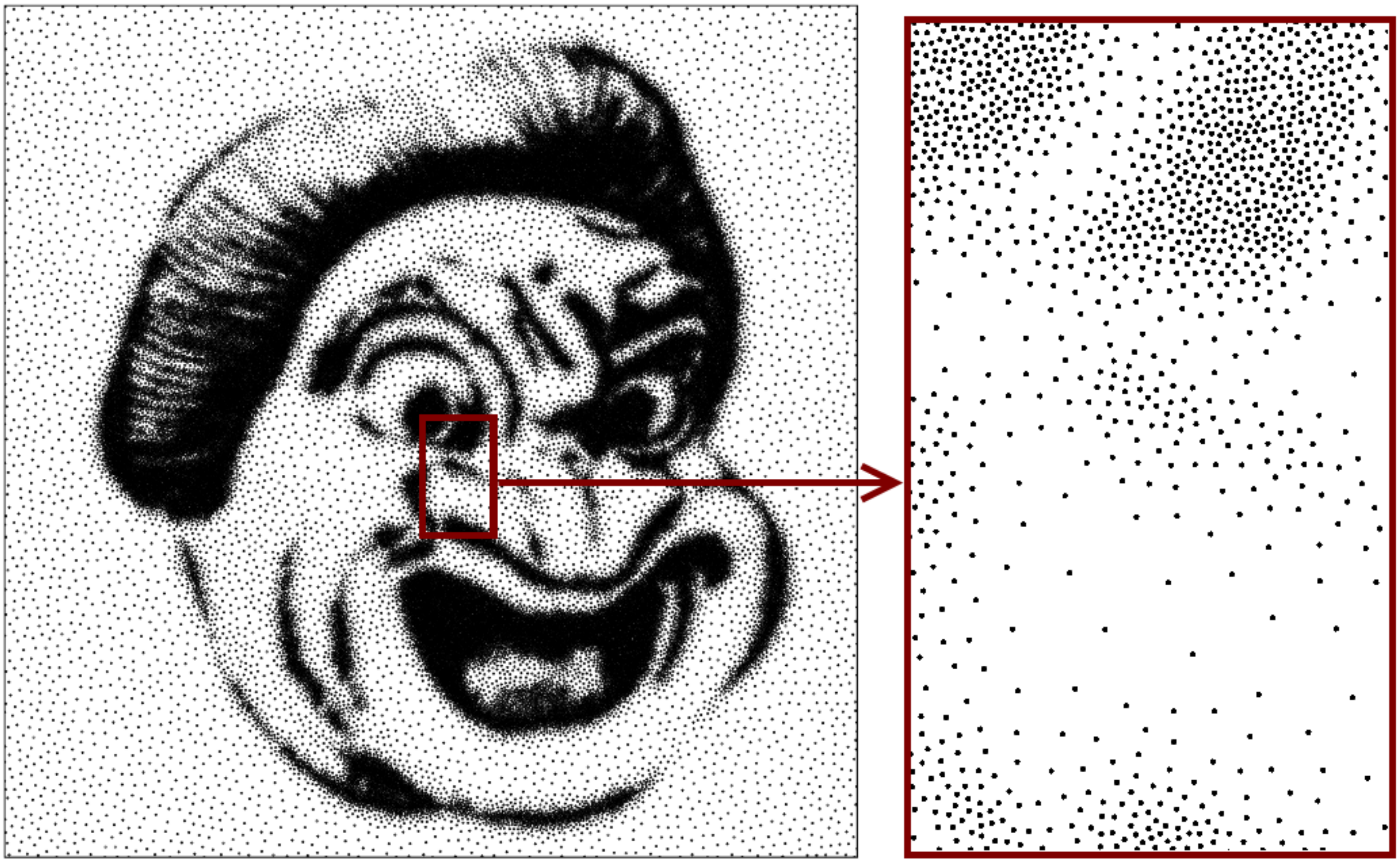
British Columbia

- Random sphere packing
  - Non-overlapping r/2 disks
  - Atoms in a liquid, crystal



Motivation from Static Graphics

- **Stippling: images from dots, as newsprint**



(Brush) Stroke-Based Rendering

- **CG artistic effect to mimic physical media**
- **Images from Aaron Hertzmann, Stroke-Based Rendering**

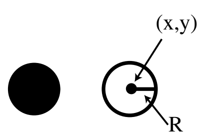
Source photo

Painted version

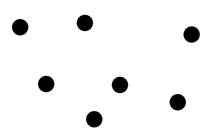
Final rendering

Definition: A **stroke** is a data structure that can be rendered in the image plane. A **stroke model** is a parametric description of strokes, so that different parameter settings produce different stroke positions and appearances.

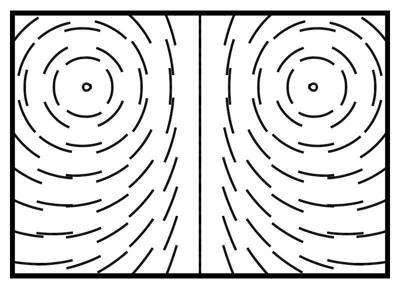
For example, one form of stippling uses a very simple stroke model:



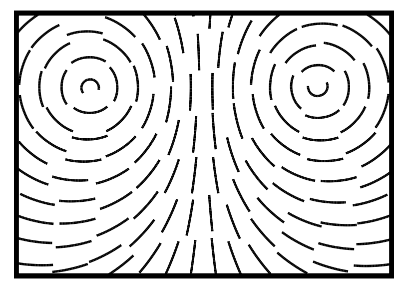
Stippling stroke model



Individual strokes (stipples)



Vector field

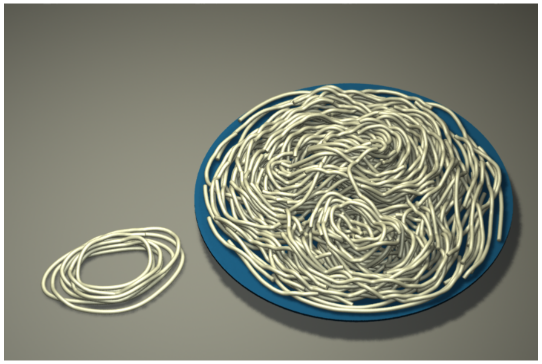


Final rendering

Sandia National Laboratories

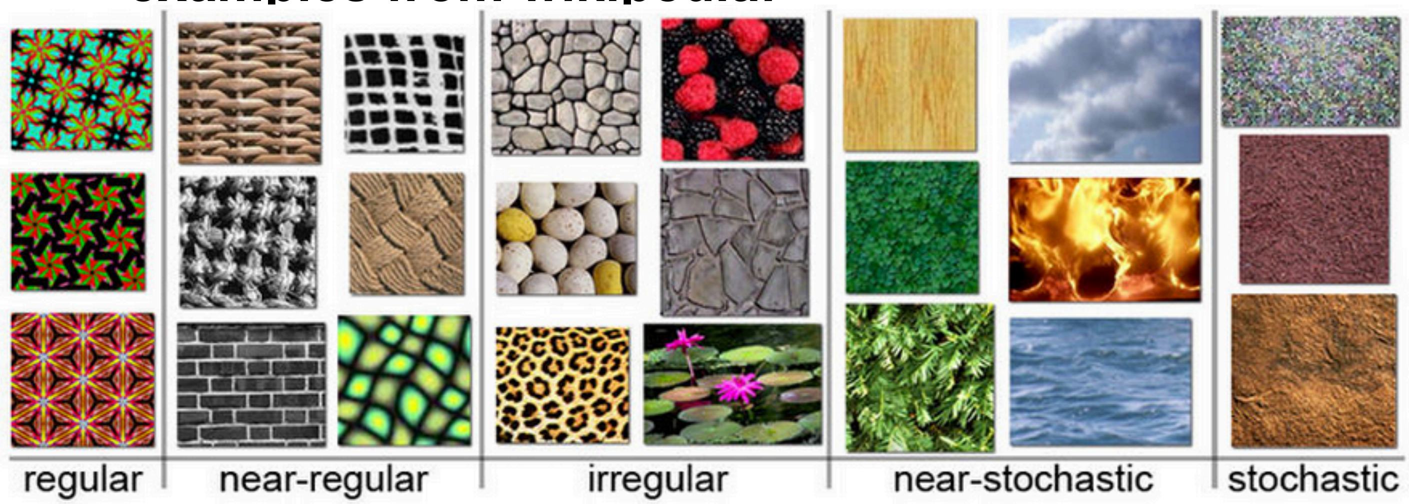
Motivating from Modern Graphics: Texture Synthesis

- Real-time environment exploration. Games! Movies!
- Algorithm to create output image from input sample
  - Arbitrary size
  - Similar to input
  - No visible seams, blocks
  - No visible, regular repeated patterns



Spaghetti
Li Yi Wei
SIGGRAPH 2011

examples from wikipedia:

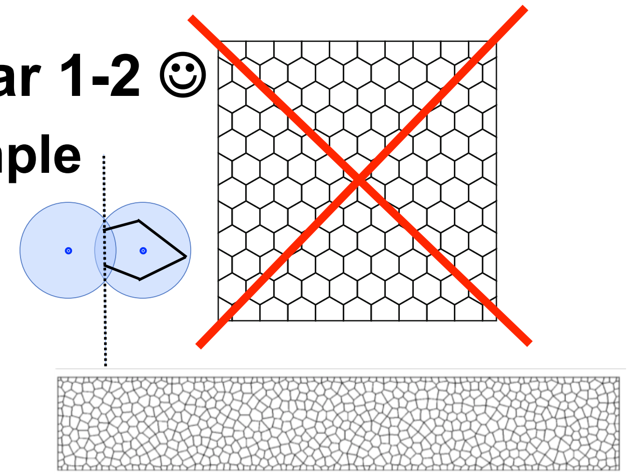


What is MPS good for?

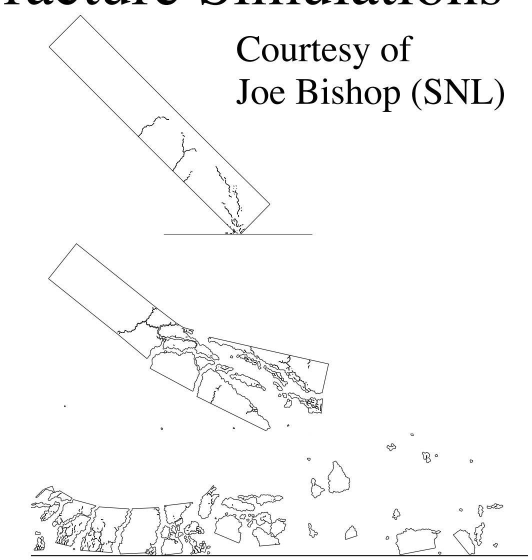
- Humans are very good at noticing patterns, even ones that aren't there
  - Patternicity: Finding Meaningful Patterns in Meaningless Noise, Scientific American Dec 2008
  - Cognition issues…side exploration
- Our eyes sensitive to patterns
- Randomness hides imperfections
  - stare at dry-wall in your house sometime, try to find the seams
- Unbiased process leads to points with
  - No visible patterns between distant points.
    - pairwise distance spectrum close to truncated blue noise powerlaw

What is MPS good for?
Sandia cares about Games and Movies? training…

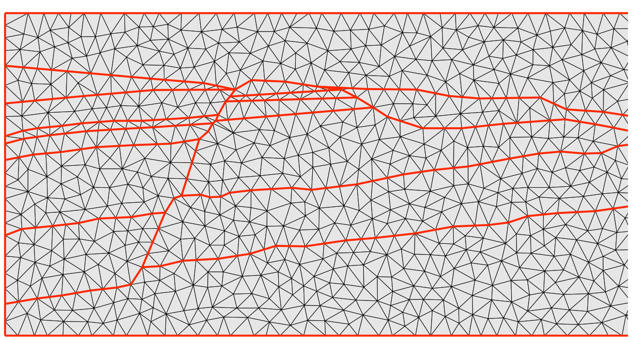
- Physics simulations – why SNL paid for year 1-2 ☺
  - Voronoi mesh, cell = points closest to a sample
  - Fractures occur on Voronoi cell boundaries
    - Mesh variation ⊂ material strength variation
    - CVT, regular lattices give unrealistic cracks
      - Unbiased sampling gives realistic cracks
  - Ensembles of simulations
  - Domains: non-convex, internal boundaries



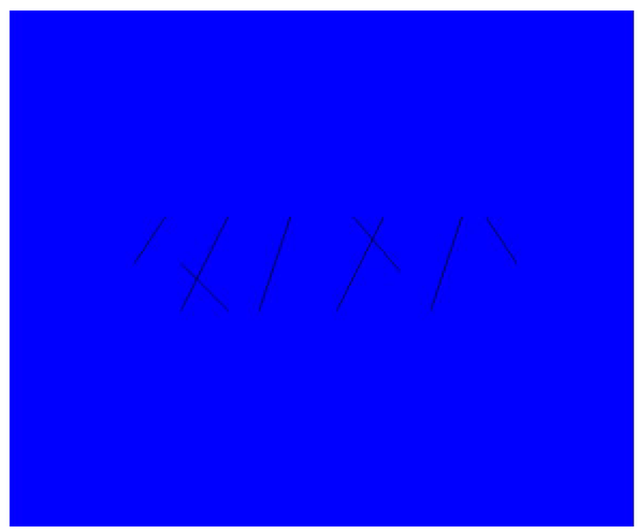
Fracture Simulations



Courtesy of
Joe Bishop (SNL)



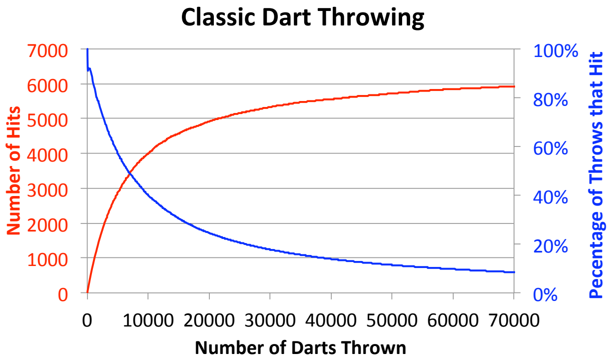
Seismic Simulations
maximal helps Δ quality

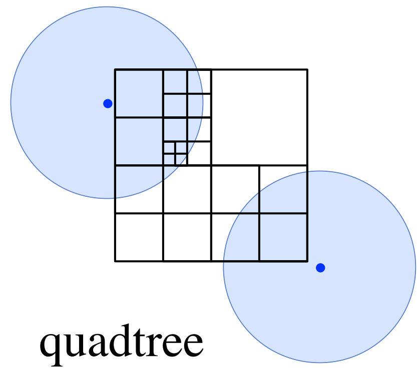


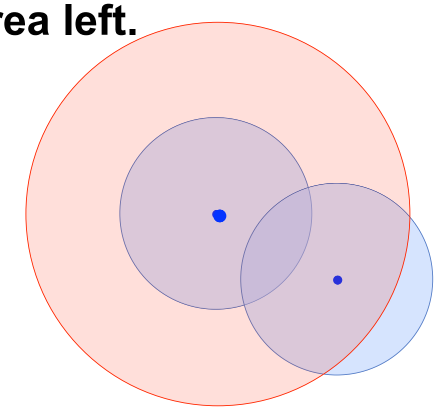
Sandia National Laboratories

Algorithm for MPS

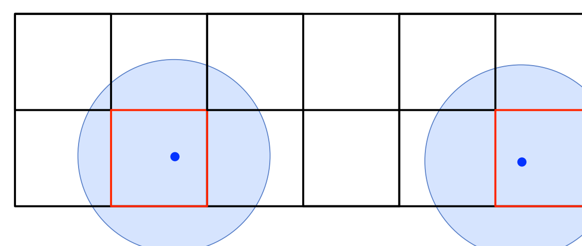
- **Classic algorithm**
  - **Throw a point, check if disk overlaps, keep/reject**
  - **Fast at first, but slows due to small uncovered area left. Can't get maximal.**




quadtree


advancing front

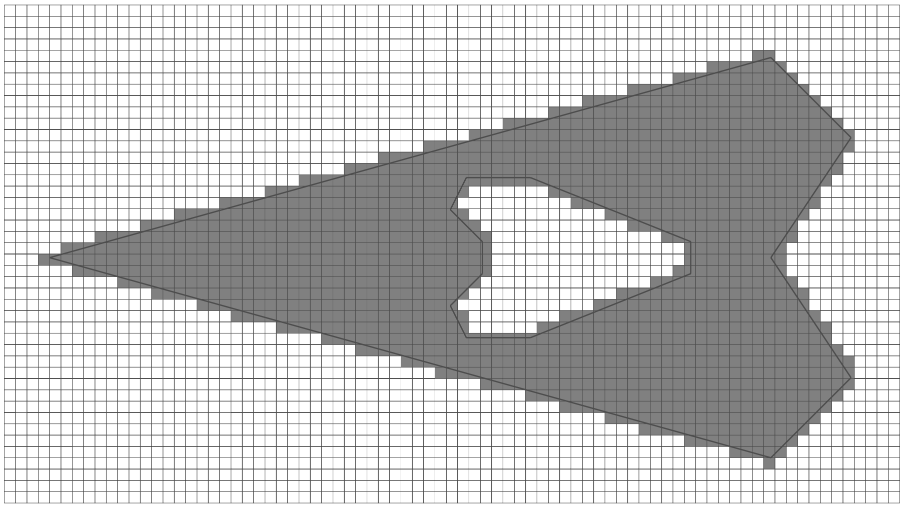
- **Speedup by targeting just the uncovered area**
  - **Others use quadtrees to approximate the uncovered area**
  - **Others use advancing front to sample locally**
  - **Others use tiles to aid parallelism**


independent tiles

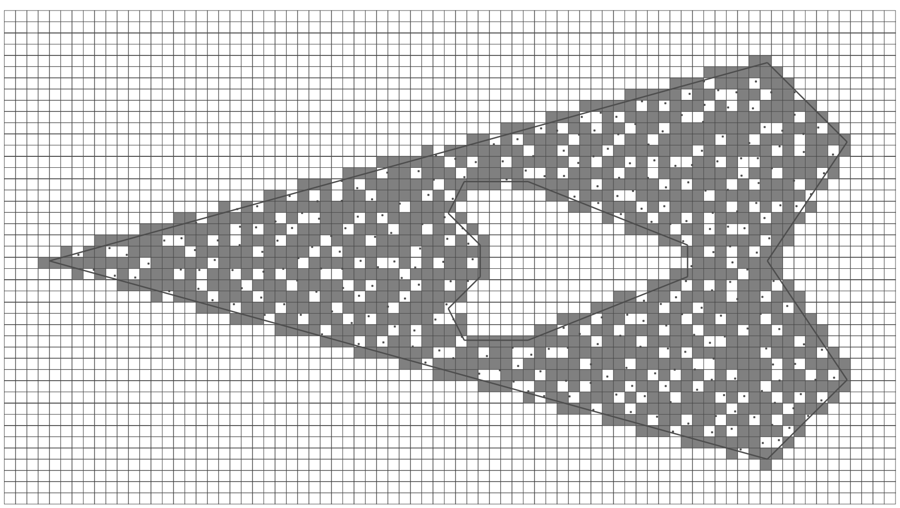
- **Common issues**
  - **Not strictly "unbiased" process**
    - **Outcome may be indistinguishable from an unbiased process's outcome**
  - **Not maximal: dependent on finite precision**
  - **Memory or run-time complexity**
  - **Ours is first provably bias-free, maximal, E(*n* log *n*) time O(*n*) space**

Sandia National Laboratories

Algorithm

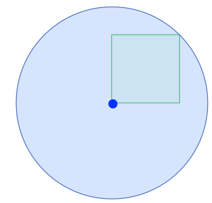


Initial Pool C



End of Phase I: white cells with a point

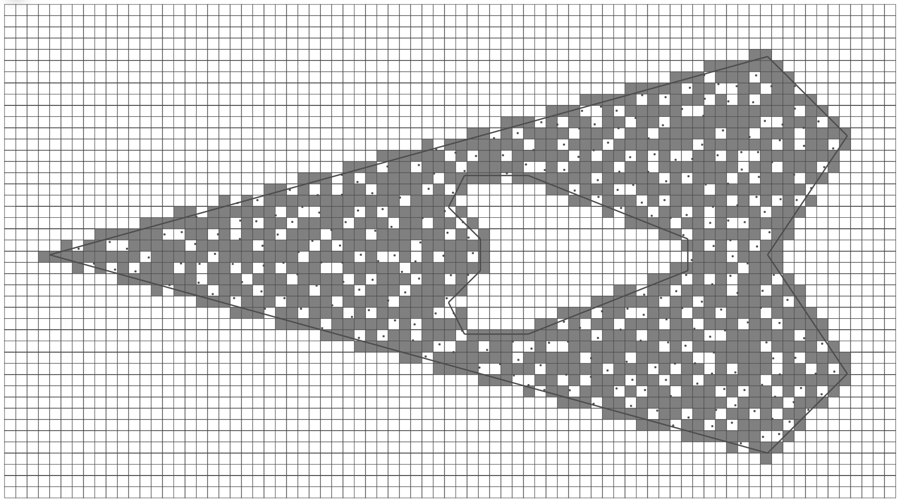
- Background square grid
  - Square diagonal = r



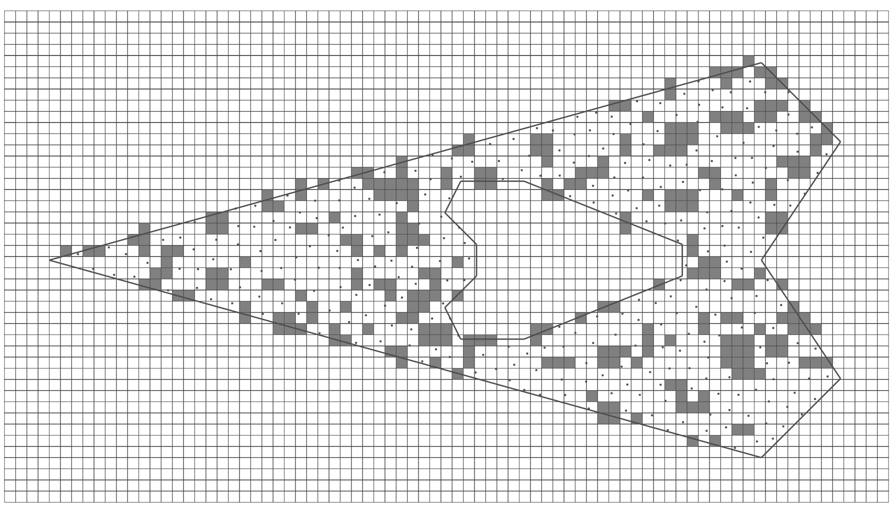
- Flood fill
  - Build pool of cells C : not-exterior to domain
- Phase I: quickly cover most of the domain
  - Pick a square from pool
  - Pick point in square
  - If point uncovered (likely)
    - Keep point
    - Remove square from pool
  - Repeat $a|C|$ times

Sandia National Laboratories

Algorithm

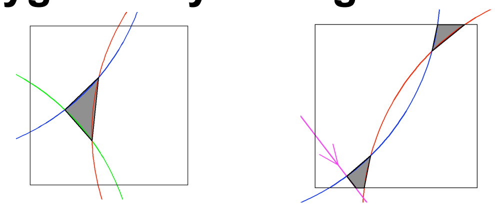


End of Phase I: white cells with a point



Start of Phase II: dark cells not-covered

- Target remaining uncovered area
- Construct square \ disks
  - Polygon easy surrogate for arc-gon



- Replace pool of squares by polygons
- Phase II: repeat
  - Pick polygon from pool
    - Weighted by its area (only log n step)
  - Pick point in polygon
  - If uncovered
    - Keep point
    - Remove polygon from pool
    - Update nearby polygons
- Works well because
  - Voids are scattered
  - Small arc-gons are well approximated by polygons

Sandia National Laboratories

Algorithm Nuance - Phase II stages

- "Algorithm is simple,… in a good way" - Reviewer
- **Lazy update of polygons' areas and pool, in "stages"**
  - More simple datastructures
  - No tree needed, flat array for pool, fewer pointers
  - Run-time proof gets more complicated

Prior slide

```
Phase II: repeat
    Pick polygon from pool
        Weighted by its area (only log n step)
    Pick point in polygon
    If uncovered
        Keep point
        Remove polygon from pool
        Update nearby polygons
```

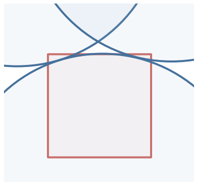
→

Lazy update

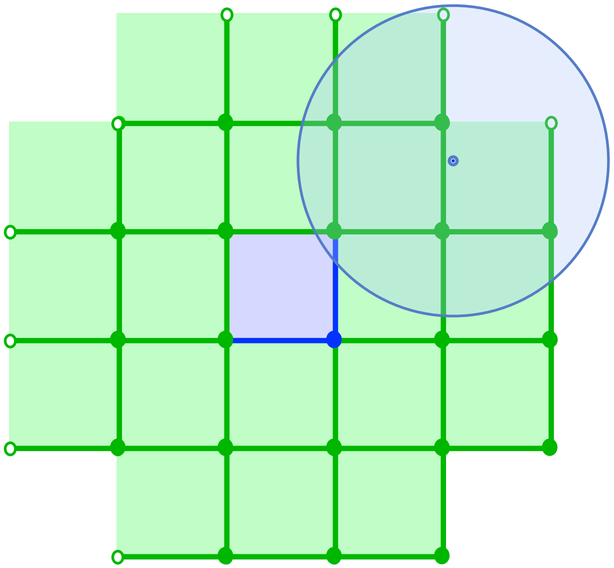
```
Phase II: repeat
  Repeat c|Pool| times
    Pick polygon from pool
        Weighted by its area (only log n step)
    Pick point in polygon
    If uncovered
        Keep point
  New stage - update all polygons
  Rebuild pool and weights
```

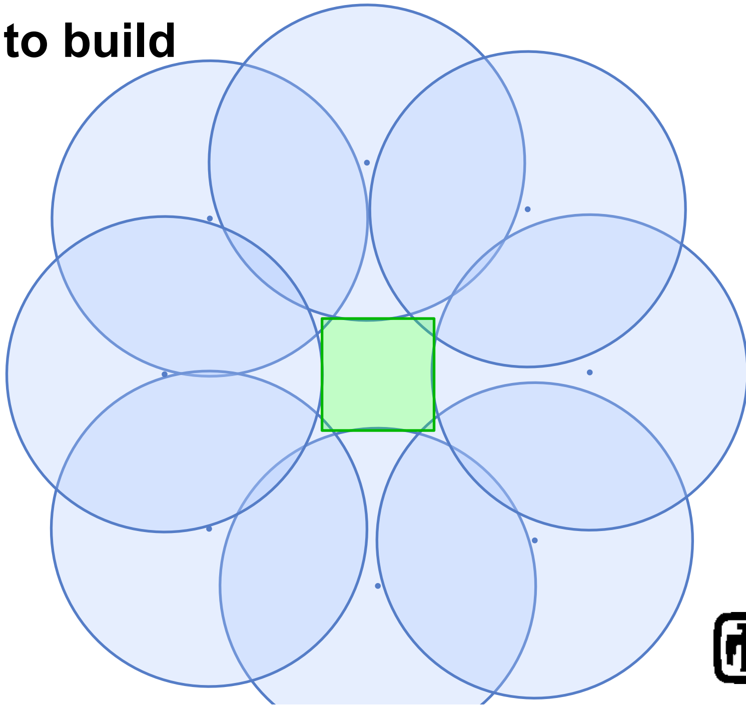
Complexity Proofs Sketch

- WTS constant time & space per point
  - Everything is local, and constant size
- #squares = θ(#points_in_sample)
- Sid Meier Civilization template
  - 21 nearby squares, 0 or 1 disks per square
    - By geometry, ≤ 4 voids per cell
    - By geometry, ≤ 9 (8?) disks bounding a void
- Constant time to check if point is uncovered
- Polygons are constant size, time to build



Four voids





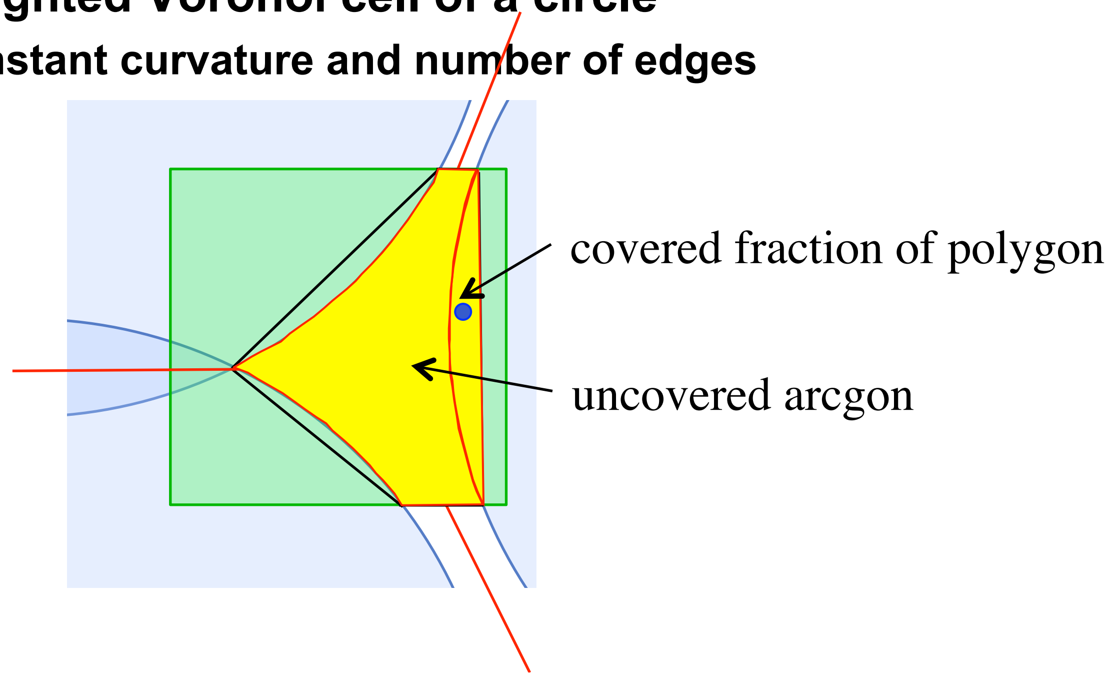
Complexity Proofs Sketch

- **Constant work per generated point, but what about the rejected (covered) points?**
  - **Phase I, O(|C|) throws**
  - **Phase II**

$$\text{Area(arcgon)} > c\,\text{Area(polygon)} \Leftrightarrow P(x_i : \text{uncovered}) > c$$

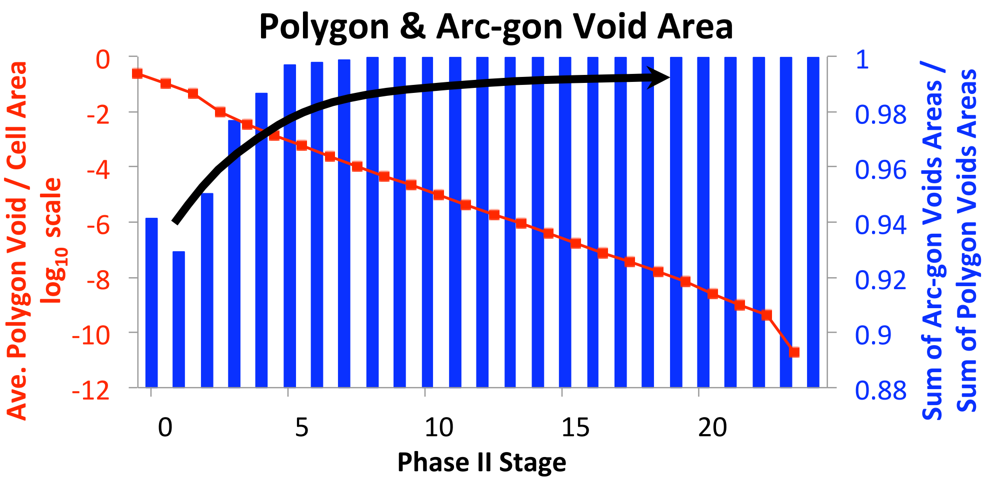
$$\Leftrightarrow \#\,\text{accepted} > c_2\,\#\,\text{rejected}$$

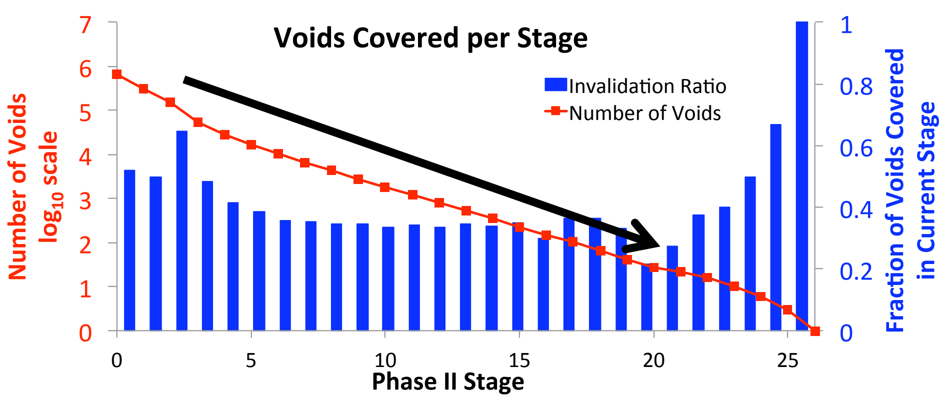
  - **Via weighted Voronoi cell of a circle**
    - **Constant curvature and number of edges**

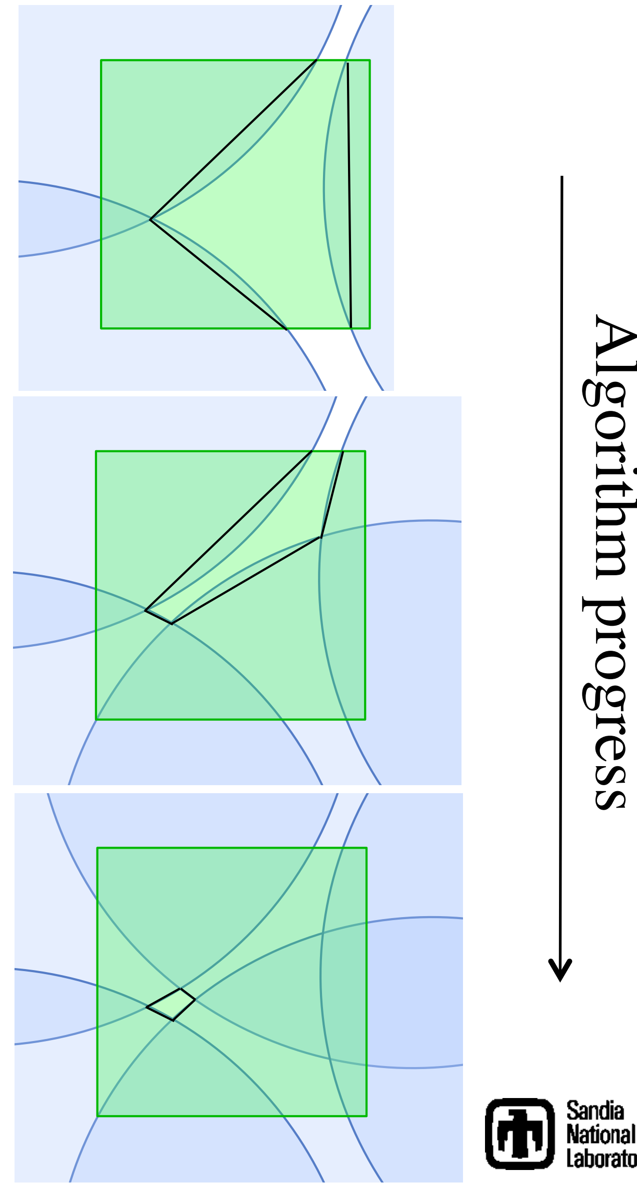


Fewer Rejected Points Later

- **Polygons ➔ arcgon as voids get smaller**
  - **We get more efficient (contrast)**







Sandia National Laboratories

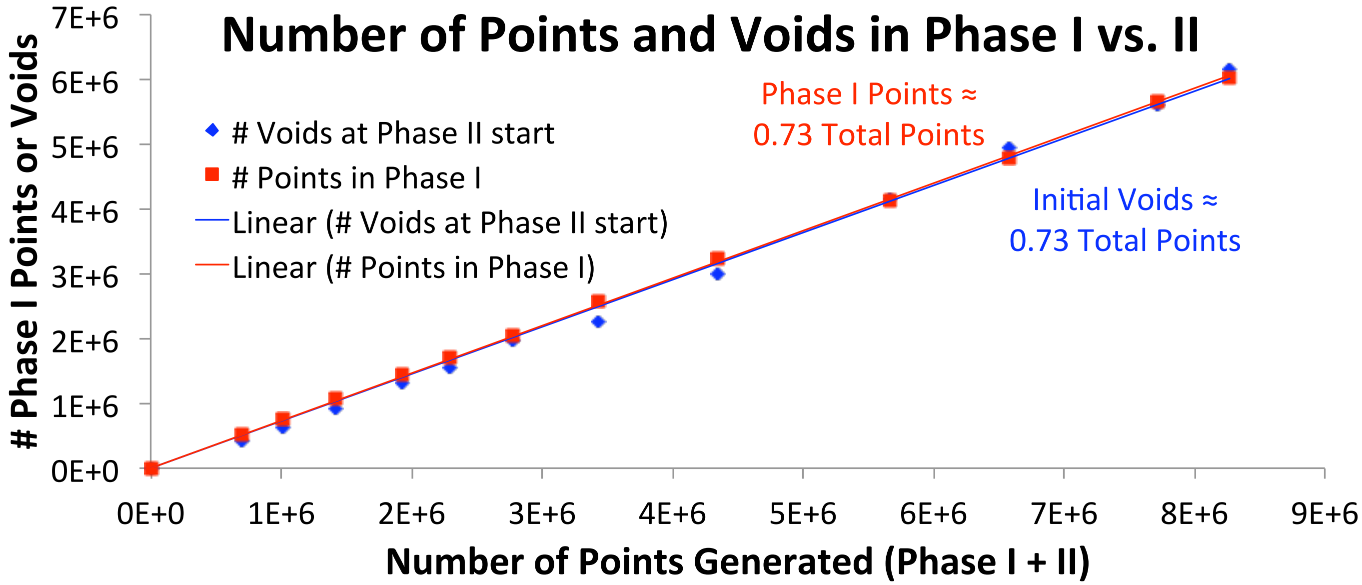
Complexity

- Complexity – everything is local, all steps constant time
  - except log(n) to select a polygon, weighted by area
  - that is a relatively inexpensive step
  - constructing geometric primitives is the expensive part
- Constant fraction of generated points are output points

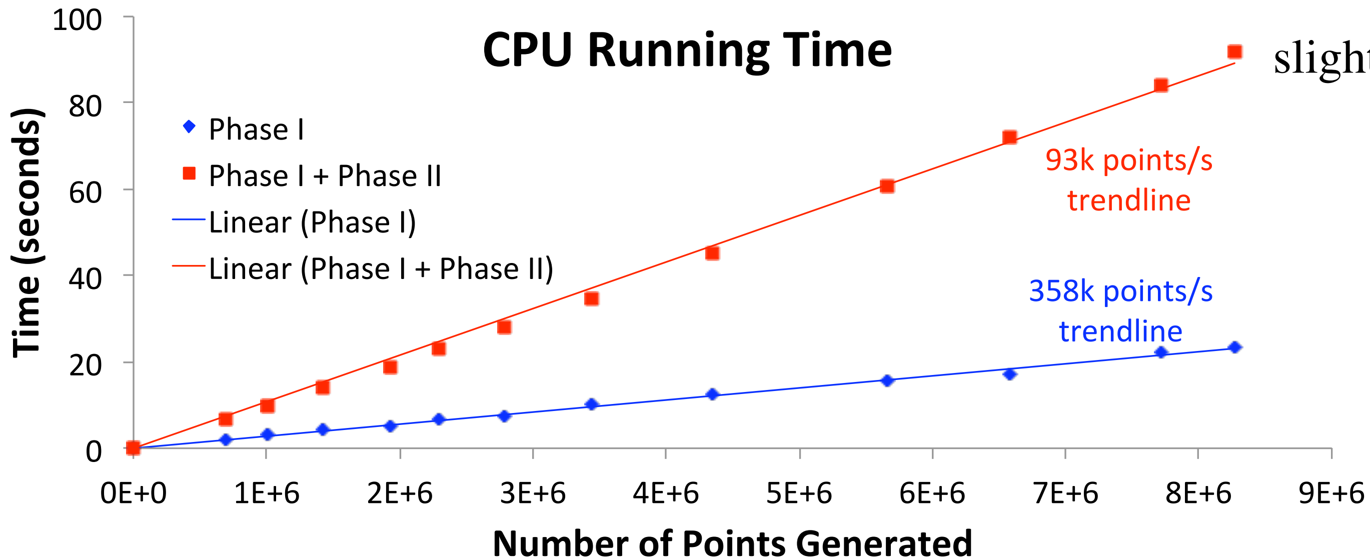
$$\mathrm{Time} = \mathrm{E}(Cn + cn\log n)$$

$$\mathrm{Space} = O(n)$$

Runtime – Why we do Phase I

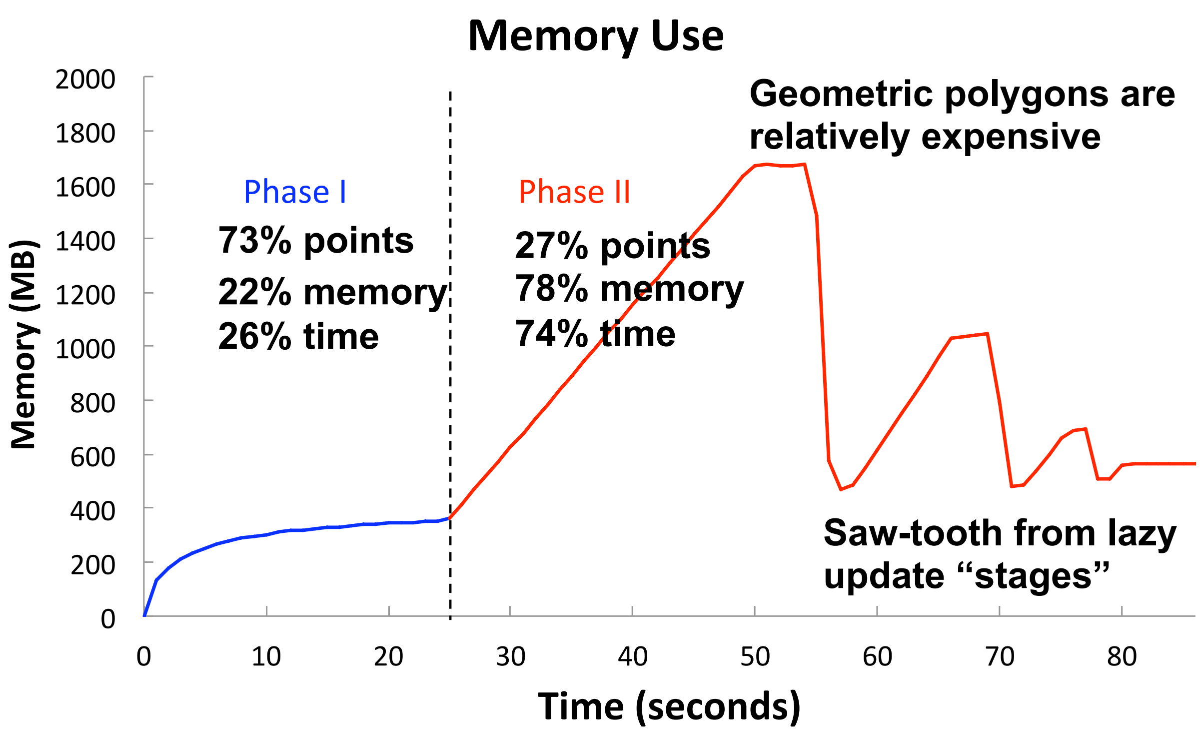


- **Phase I**
  - **73% of points**
  - **26% of runtime**



slight uptick from log

Serial Memory Use

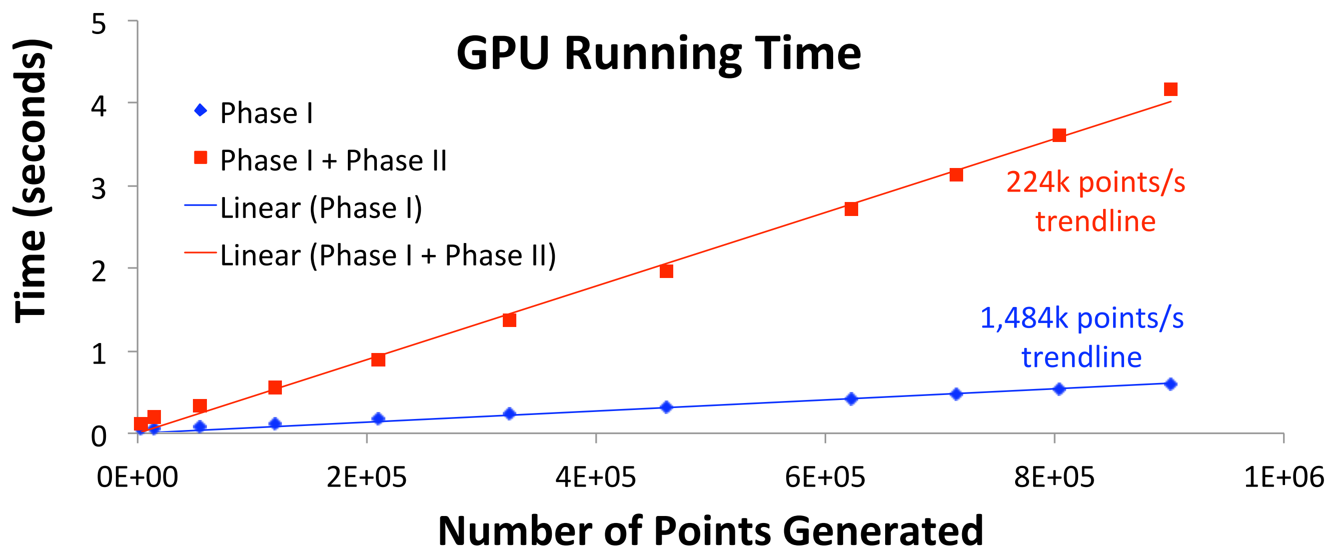


GPU Algorithm

Points generated in parallel, conflicts resolved in an unbiased way

- **Point buffers: candidate and final**
- **Phase I**
  - **Iterate: synchronize at start of iteration**
    - **Generate $|C|/5$ candidate points**
    - **Square states: empty, test, accepted, done**
      - **Done = Point from prior iterations**
      - **Test = Point doesn't conflict with nearby "done" points, compute in parallel**
      - **Accepted = Point is earlier (id) than conflicting "test" points, compute in parallel**
    - **Migrate accepted points to done, otherwise remove**
- **Phase II**
  - **Construct polygons, compute in parallel**
    - **Squares "rejected" if covered by prior disks, has no polygon, no work to do**
    - **Split polygons into triangles**
  - **Proceed as Phase I, with triangles playing role of squares**

GPU Performance

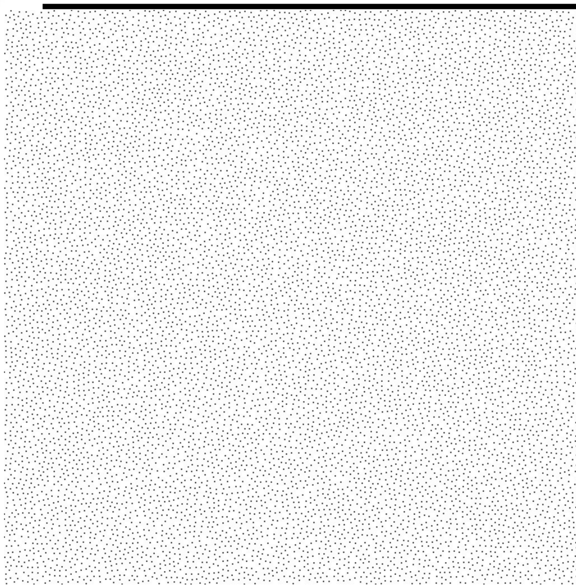


NVIDIA GTX 460

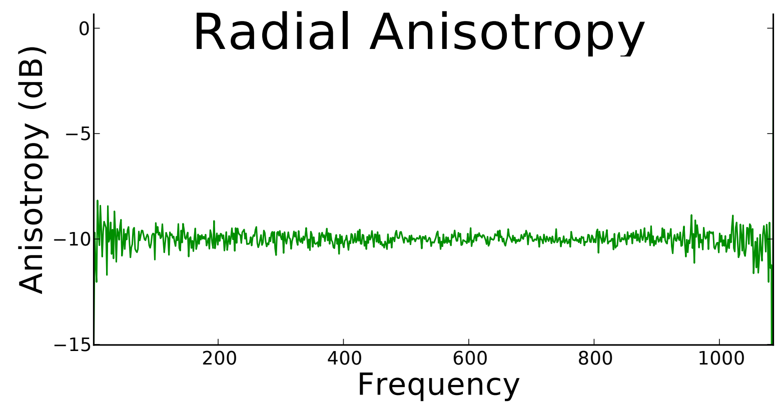
2.4x speedup over serial (6.7x memory bandwidth)

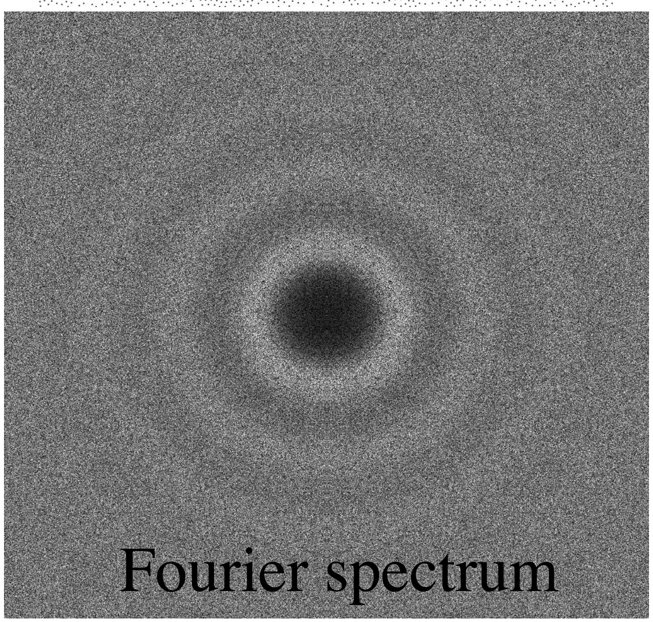
1 million points in 1 GB RAM

Unbiased Parallel Sample

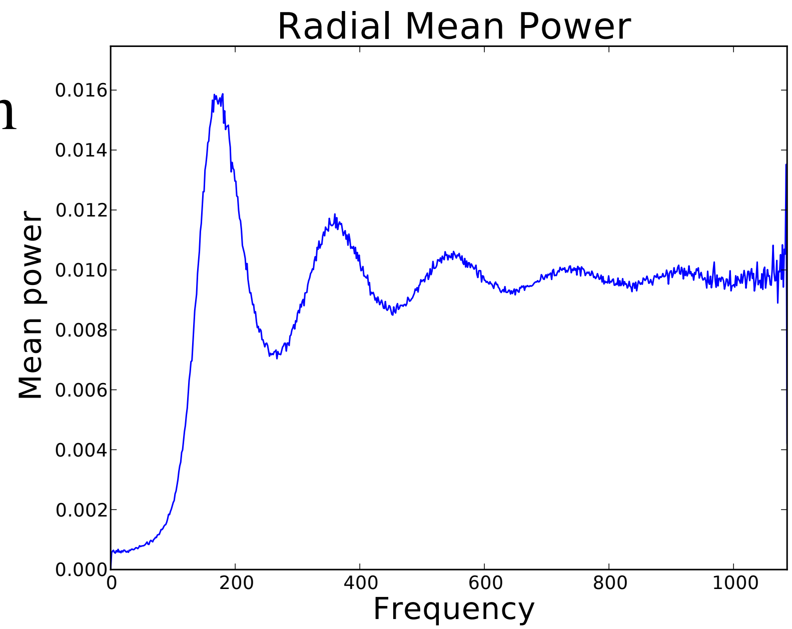


10k pts





Rings from inhibition radius

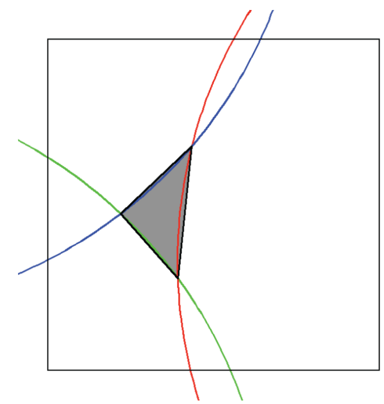


Synopsis of Contribution

- Poisson-disk distributions
  - Simple, efficient implementation
  - Provable guarantees
    - Maximal
    - Unbiased
    - O(n) space
    - $\mathrm{E}(Cn + \epsilon n \log n)$ time
- Domains
  - 2d
  - Polygons with holes, non-convex
- Algorithmic innovations
  - Two phases

    I. fast to cover most of domain

    II. careful to cover remainder
  - Approximate uncovered "voids", square $\cap$ circles, with polygons. Careful weighting and selection

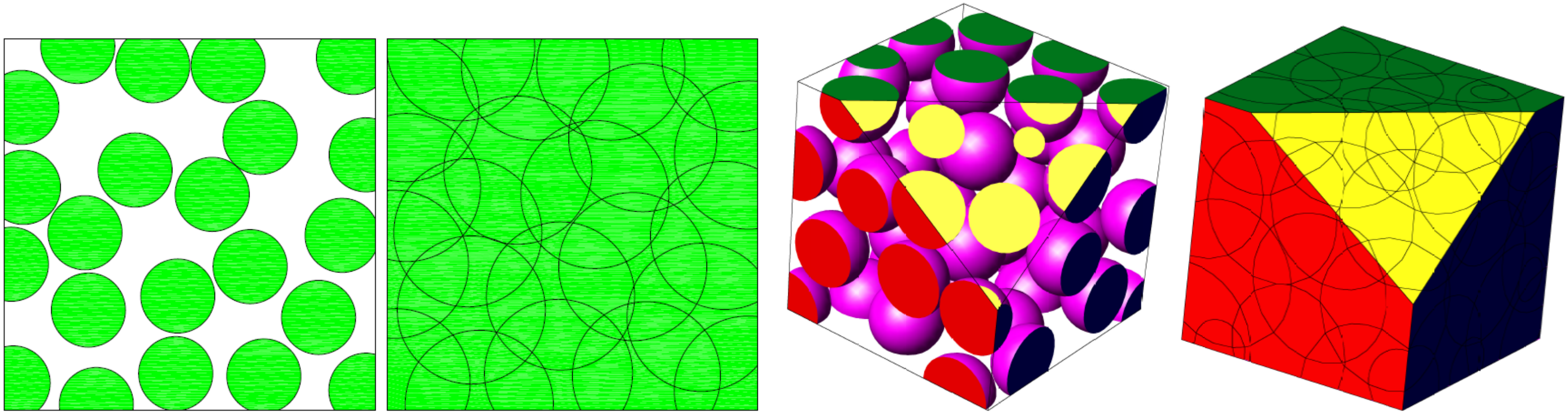


Sandia National Laboratories

Future

- Extensions
  - Could do away with polygonal approximation and weight and sample directly – every dart is a hit! (w/ Thouis Ray Jones)
- Higher dimensions
  - geometric primitives unappealing
  - prefer just use hypercubes

- Thouis Ray Jones, jgt accepted paper
  - model explicit time-of-arrival for each point
  - synchronize locally as needed
  - vs. unbiased by one dart at a time, inherently serial



A Simple Algorithm for Maximal Poisson-Disk Sampling in High Dimensions

Mohamed S. Ebeida, Scott A. Mitchell, Anjul Patney,
Andrew A. Davidson, and John D. Owens

presenter = Scott

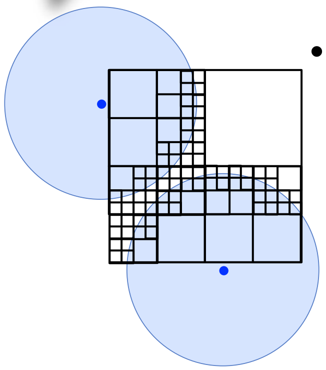
Eurographics 2012

UCDAVIS
UNIVERSITY OF CALIFORNIA

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Sandia National Laboratories

Overview



- Classic Dart throwing +
  - Quadtree
  - Squares track remaining regions
  - Track misses for refinement decisions
  - Avoid refining too deep

[Wei08] Wei L.-Y.: Parallel Poisson disk sampling.
ACM Transactions on Graphics 27, 3 (Aug. 2008), 20:1–20:9.

[BWWM10] Bowers J., Wang R., Wei L.-Y., Maletz D.:
Parallel Poisson disk sampling with spectrum analysis on surfaces.
ACM Transactions on Graphics 29 (Dec. 2010), 166:1– 166:10.

"Make everything as simple as possible, but not simpler." – A. Einstein

- Flat quadtree – one level of squares active, pool of indices
  - Simpler Datastructure ☺ Less memory ☺
- Globally refine periodically, ignore local misses
  - Simpler Datastructure ☺ More parallel ☺
- Refine to machine precision,
  on average it is so rare that memory is not an issue
  - More Maximal ☺

"This could be the current algorithm of choice for dart throwing." – Eurographics reviewer #2

- Code works great but we can't prove the spatial stats theory.

Provable:
Ebeida M. S., Patney A., Mitchell S. A., Davidson A., Knupp P. M., Owens J. D.:
Efficient maximal Poisson-disk sampling.
ACM Transactions on Graphics 30, 4 (July 2011), 49:1–49:12

Sandia National Laboratories

Maximal Poisson-Disk Sampling

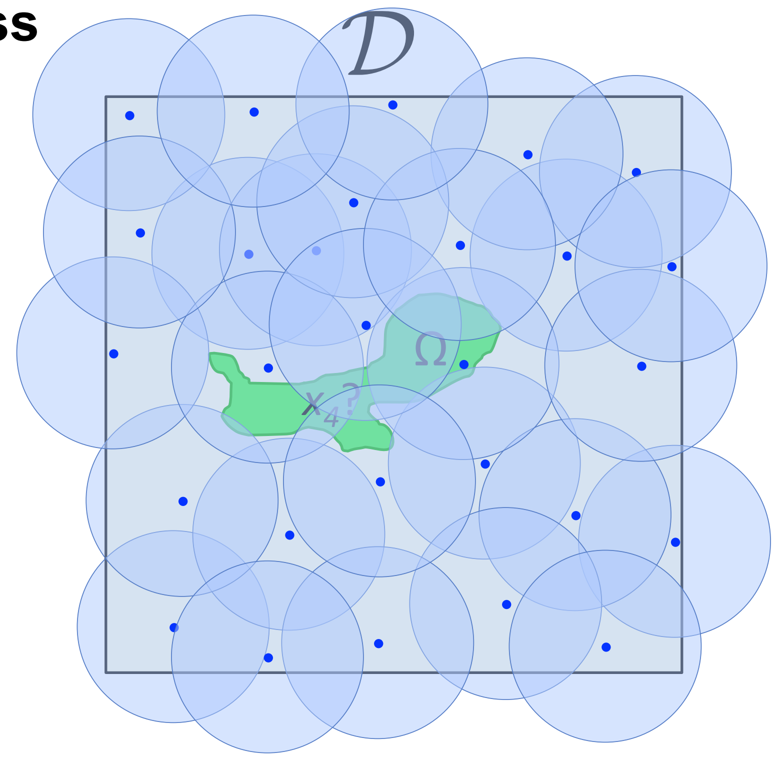
- **What is MPS?**
  - **Dart-throwing**
  - **Insert random points into a domain, build set X**
    - **With the "Poisson" process**

Empty disk: $\forall x_i, x_j \in X, x_i \neq x_j : ||x_i - x_j|| \geq r$

Bias-free: $\forall x_i \in X, \forall \Omega \subset \mathcal{D}_{i-1}$:

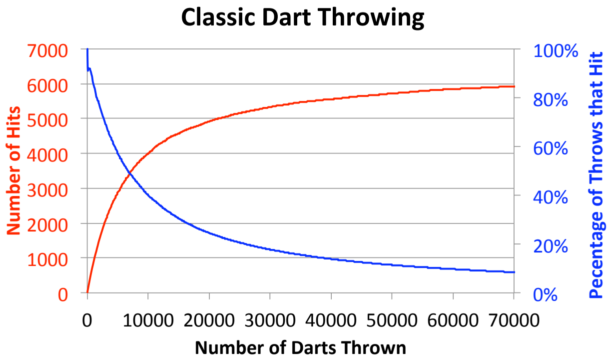
$$P(x_i \in \Omega) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$$

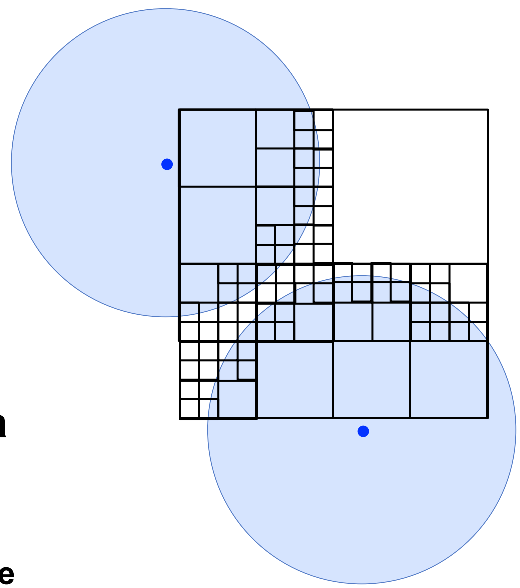
Maximal: $\forall x \in \mathcal{D}, \exists x_i \in X : ||x - x_i|| < r$



Algorithm for MPS

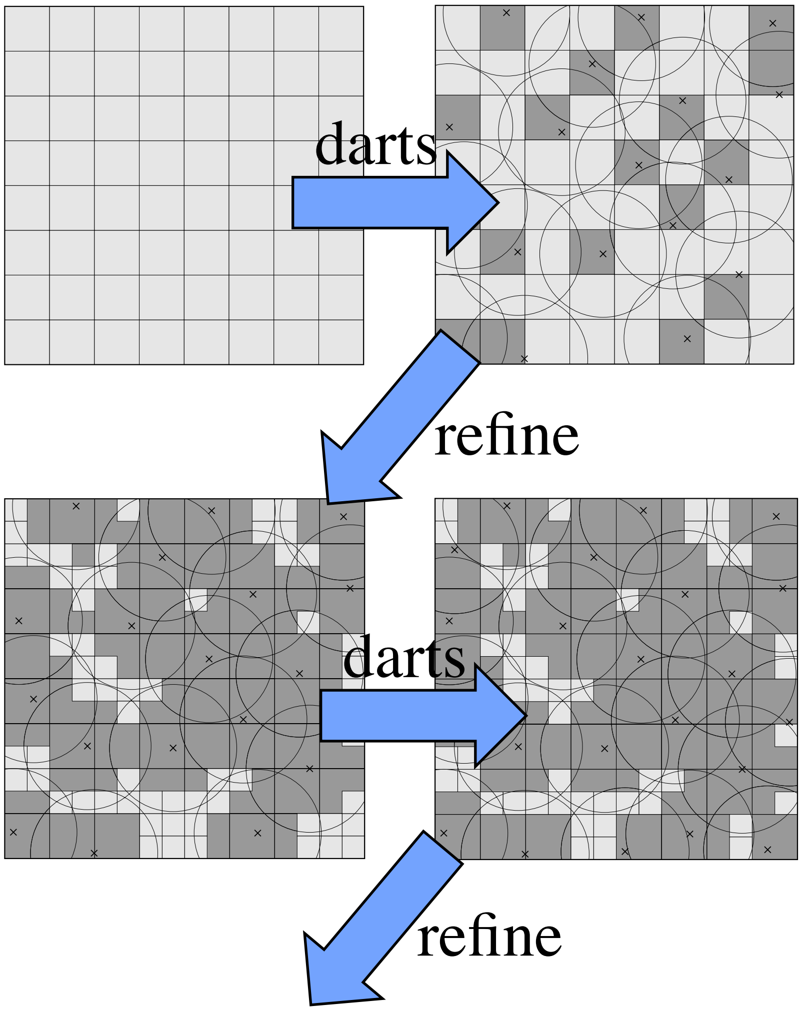
- Classic algorithm
  - Throw a point, check if disk overlaps, keep/reject
  - Fast at first, but slows due to small uncovered area left. Can't get maximal.





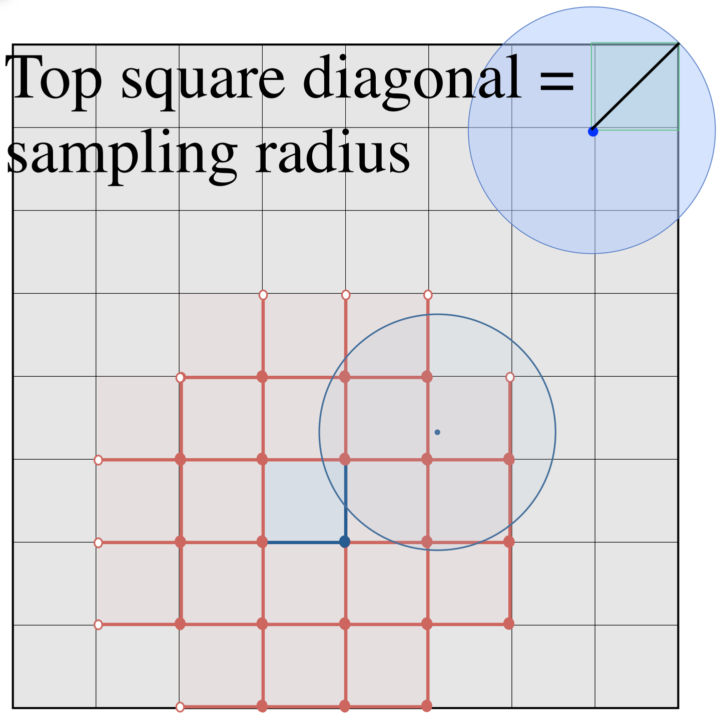
- Speedup by targeting just the uncovered area
  - Quadtrees to approximate the uncovered area
    - Discard covered squares
    - Uncovered squares: a sample is always acceptable
    - Partially covered squares: may need to refine

Our Algorithm - Basics

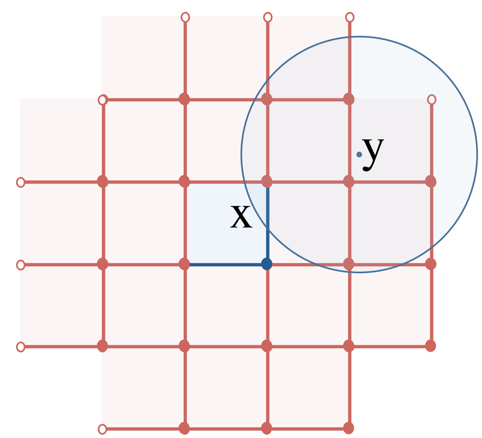


- Datastructure:
  - Squares contain uncovered area
- Throw darts
  - Pick square, pick point in square
  - If dart is outside nearby circles
    - Accept dart as sample
    - Delete square
- Refine all squares
  - Discard subsquares covered by single disks
- Repeat

Datastructure: Quadtree Root



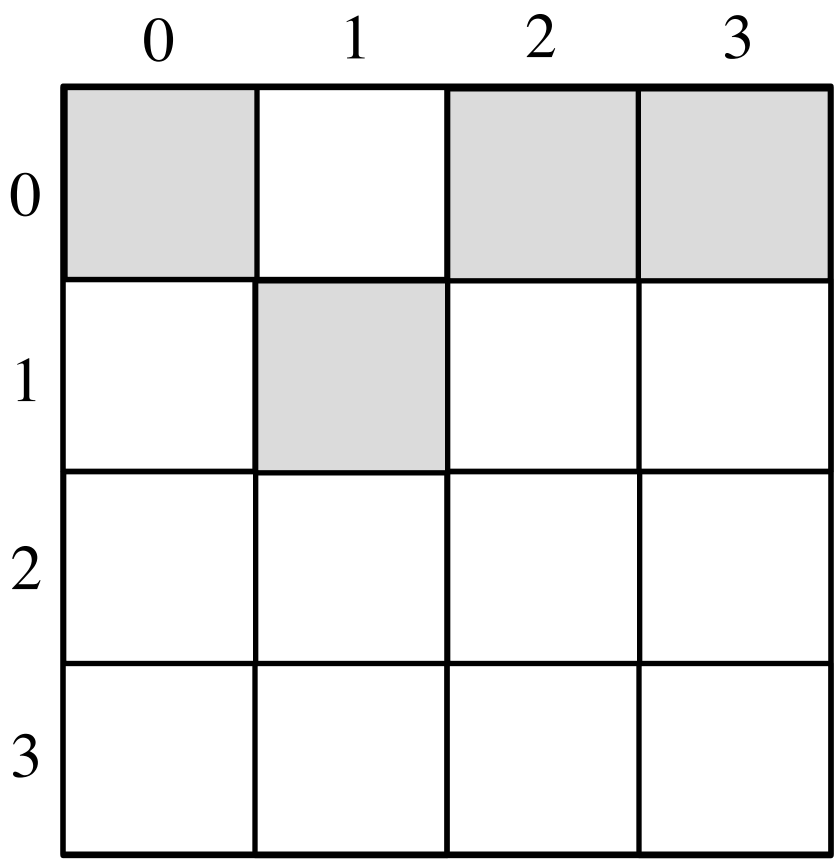
- **Squares sized so**
  - **Can fit at most one sample**
  - **Nearby square template for "Point in disk?" conflict check**
    - **Pointer from square to its sample**



Unpublished extension: use kd-tree for proximity…

Datastructure: Flat Quadtree Leaves

Flat: Only one level *i* is used at a time



- **Pool of squares**
  - **Global level *i***
  - **Squares that might accept a sample**
  - **Array of indices *C***

C^i
(0,0)
(0,2)
(0,3)
(1,1)
end
…

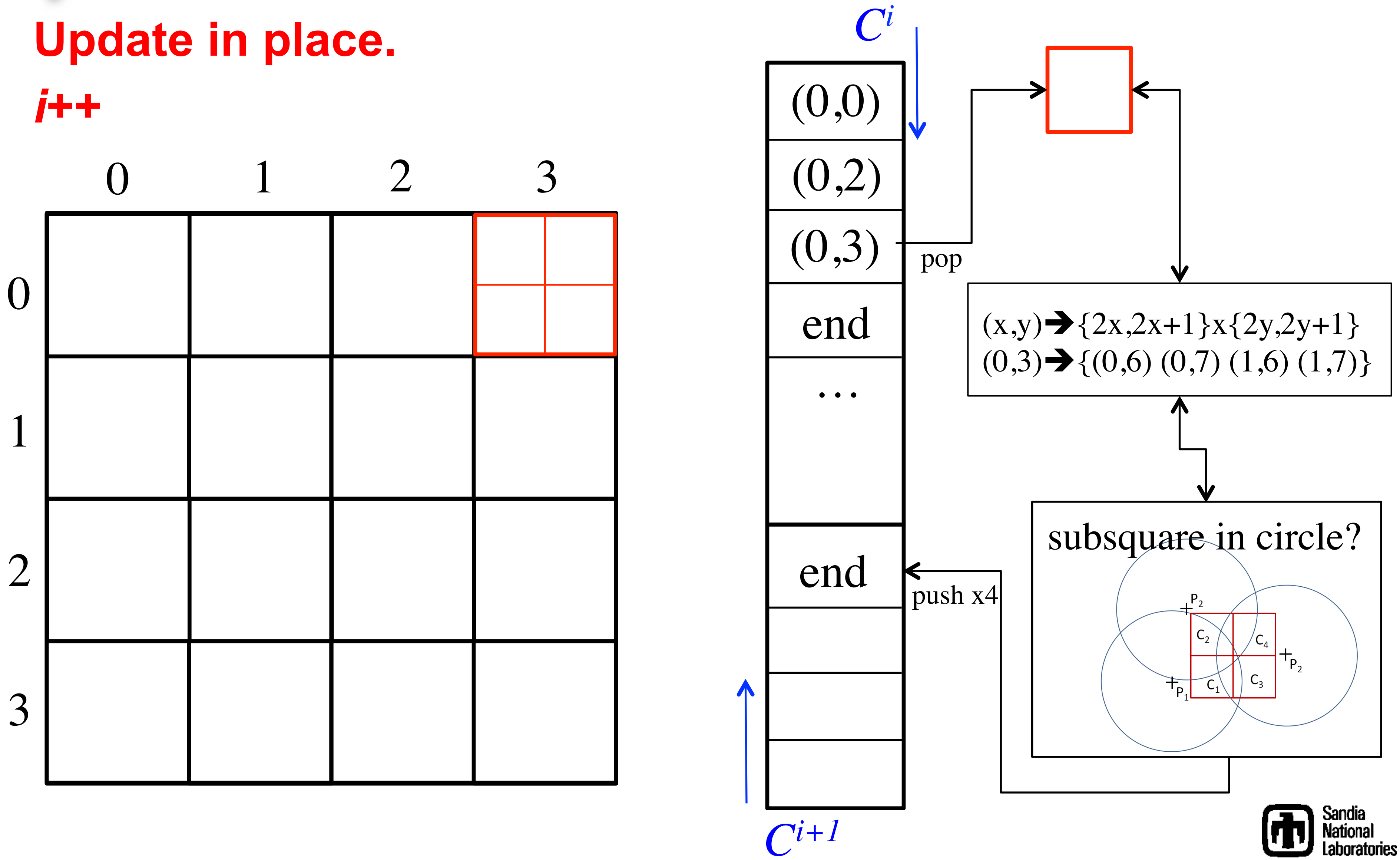
i=3
i.e. initial $\times 2^i$
squares per side

Sandia National Laboratories

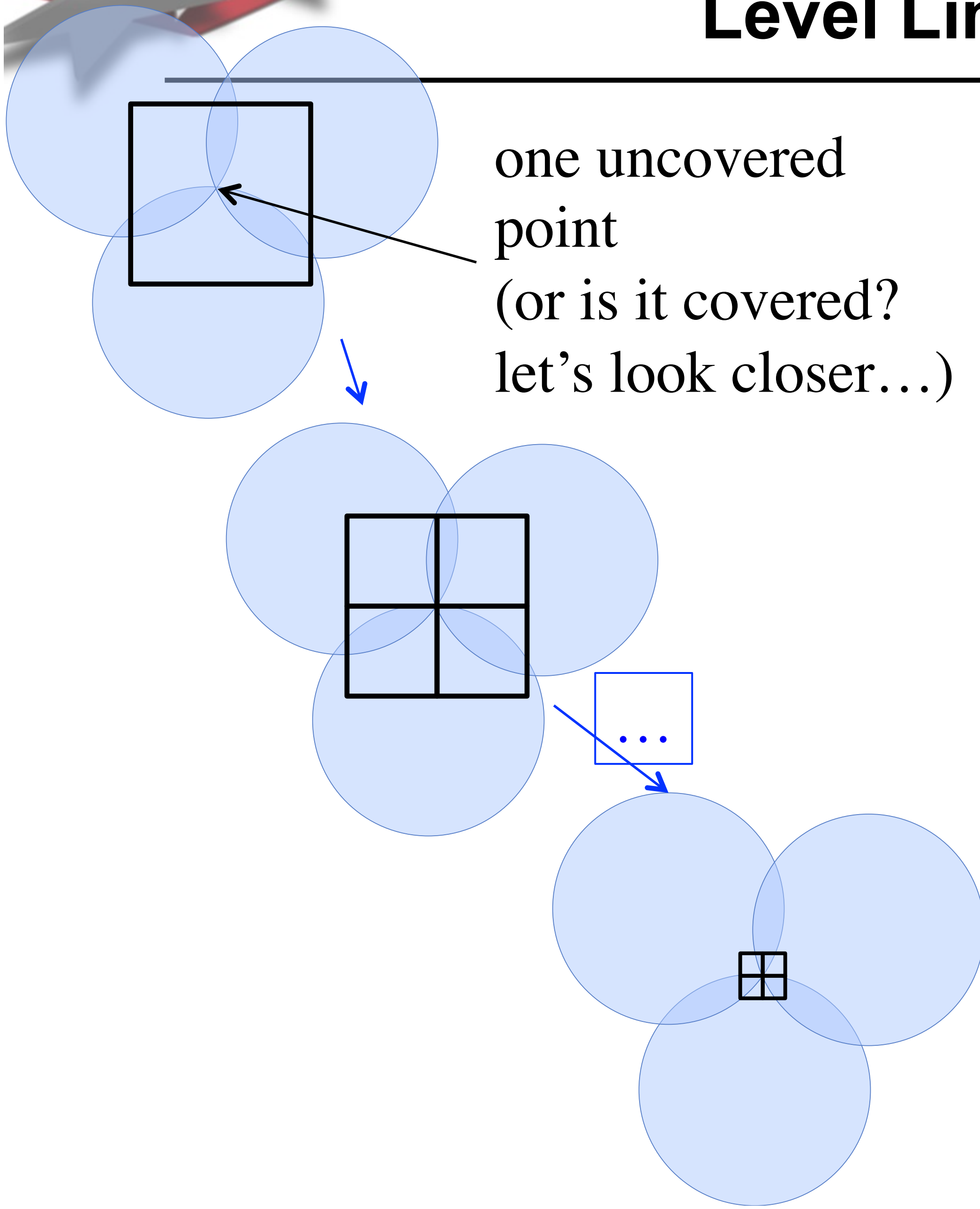
Flat Quadtree Refinement

Update in place.

i++



Level Limit?



- Problem
  - Small voids require infinite refinement
- Solution: [Wei08], [BWWM10
  - Stop early
    to avoid memory blow-up
- Solution: Us
  - Refine to finite-precision
  - Small voids happen rarely on average so
  - Memory is fine in practice
  - Benefit: maximal

Algorithm – outer loop parameters

Algorithm 1 Simple MPS algorithm, CPU.

initialize $\mathcal{G}^o$, $i = 0$, $\mathcal{C}^i = \mathcal{G}^o$
while $|\mathcal{C}^i| > 0$ **do**
  {throw darts}
  for all $\boxed{A|\mathcal{C}^i| \text{ (constant) dart throws}}$ **do**
    select an active cell $\mathcal{C}^i_c$ from $\mathcal{C}^i$ uniformly at random
    if $\mathcal{C}^i_c$'s parent base grid cell $\mathcal{G}^o_c$ has a sample **then**
      remove $\mathcal{C}^i_c$ from $\mathcal{C}^i$
    else
      throw candidate dart c into $\mathcal{C}^i_c$, uniform random
      if c is disk-free **then**
        {promote dart to sample}
        add c to $\mathcal{G}^o_c$ as an accepted sample p
        remove $\mathcal{C}^i_c$ from $\mathcal{C}^i$ {additional cells might be covered, but these are ignored for now}
      end if
    end if
  end for
  {iterate}
  for all active cells $\mathcal{C}^i$ **do**
    if $i < b$ subdivide $\mathcal{C}^i_c$ into 2^d subcells
    retain uncovered (sub)cells as $\mathcal{C}^{i+1}$
  end for
  increment i
end while

Tuning parameter choices: A, B

C^o = number initial cells

C^i = number current squares

How many throws before refining?

Throws = A | C^i |

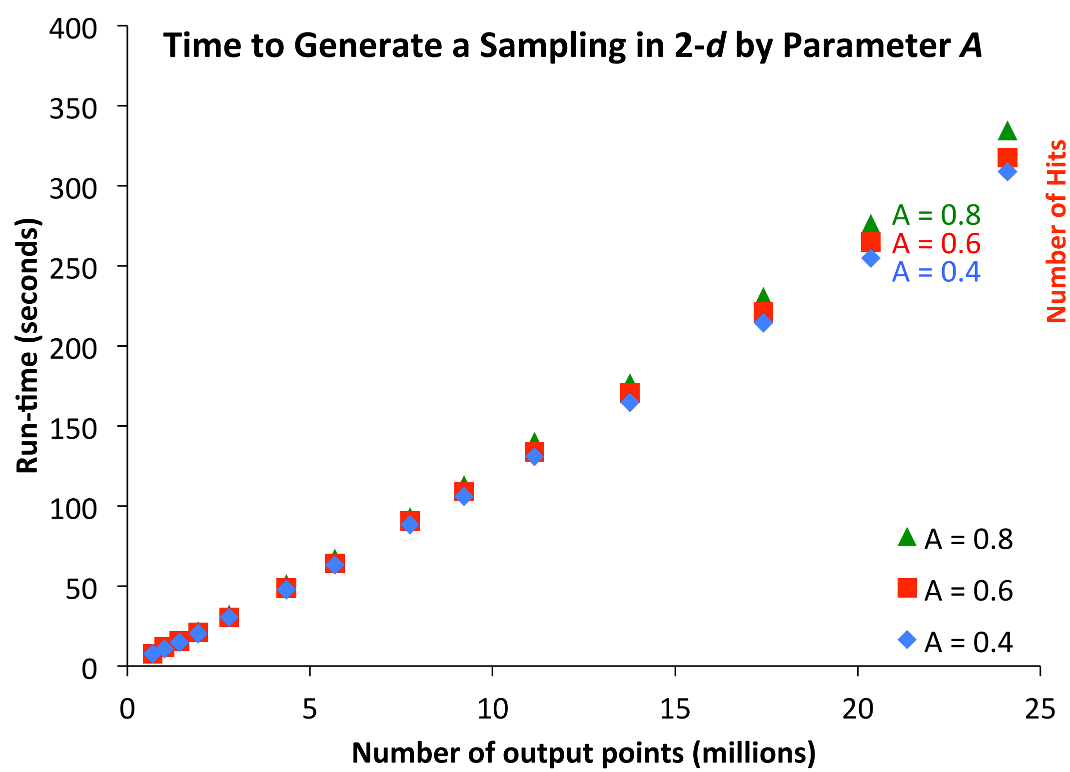
How big does array C need to be to hold all the refined grid cells?

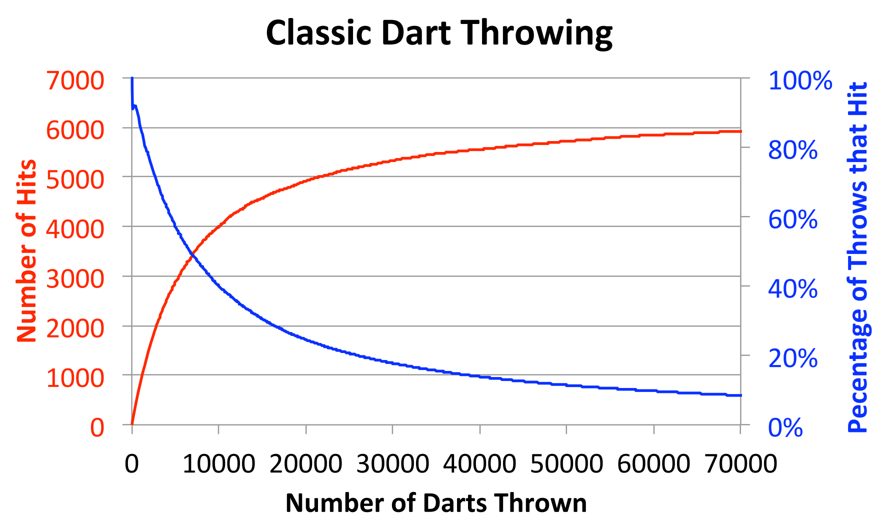
C = B | C^o |

Big A ←→ more time, smaller memory B

A (time) and B (memory) parameters

- **Big A ←→ more time, smaller memory B**
  - **A≈1, B≈dimension. (A increases for d>4)**
  - **Insensitive to value of A above a threshold**
    - **Intuition: as classical dart throwing,
      most hits happen early, no benefit to more throws**

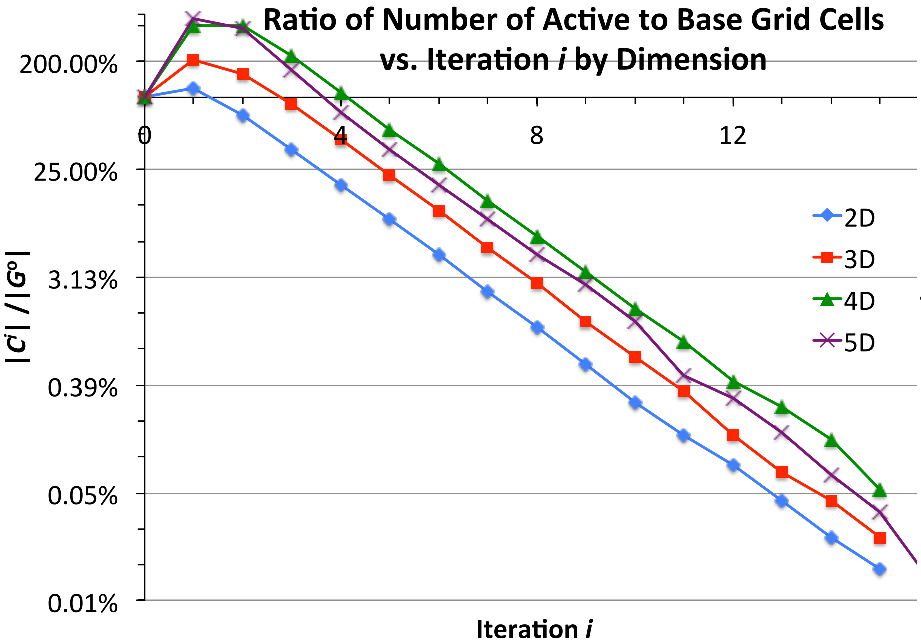


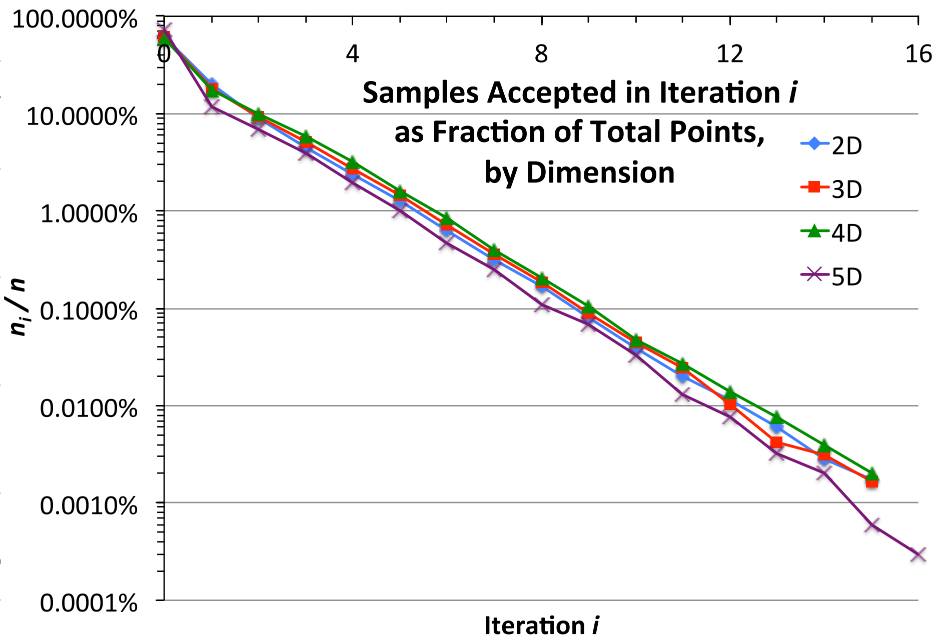


Sandia National Laboratories

Time and Memory Experimental results

- Memory and time peaks in early interations
  - Exponential convergence thereafter
  - Log y scale





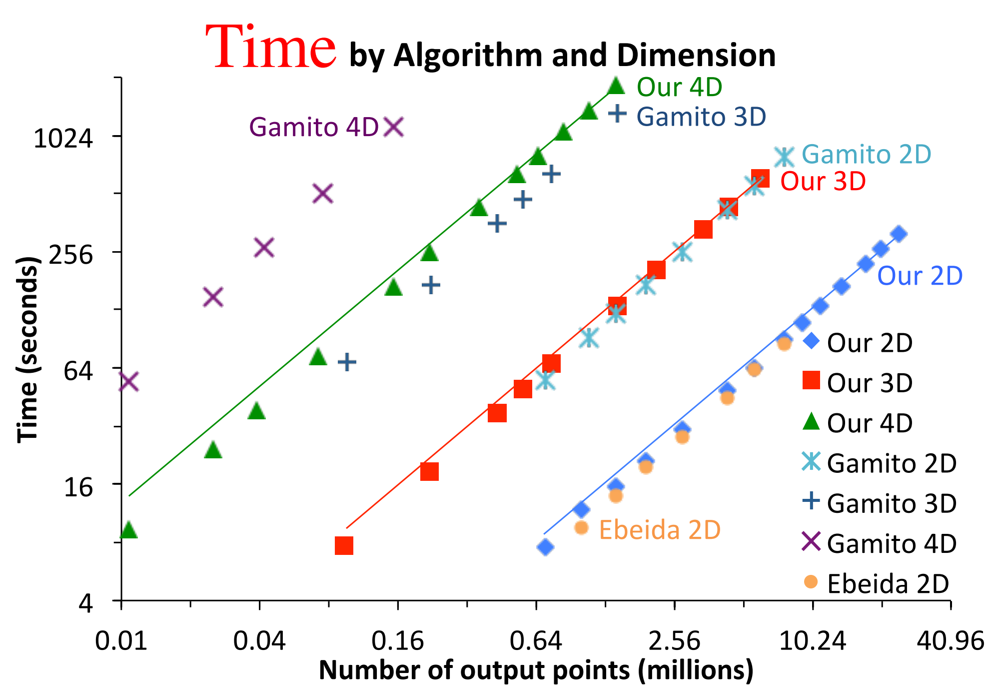
#boxes ≈ time, memory,

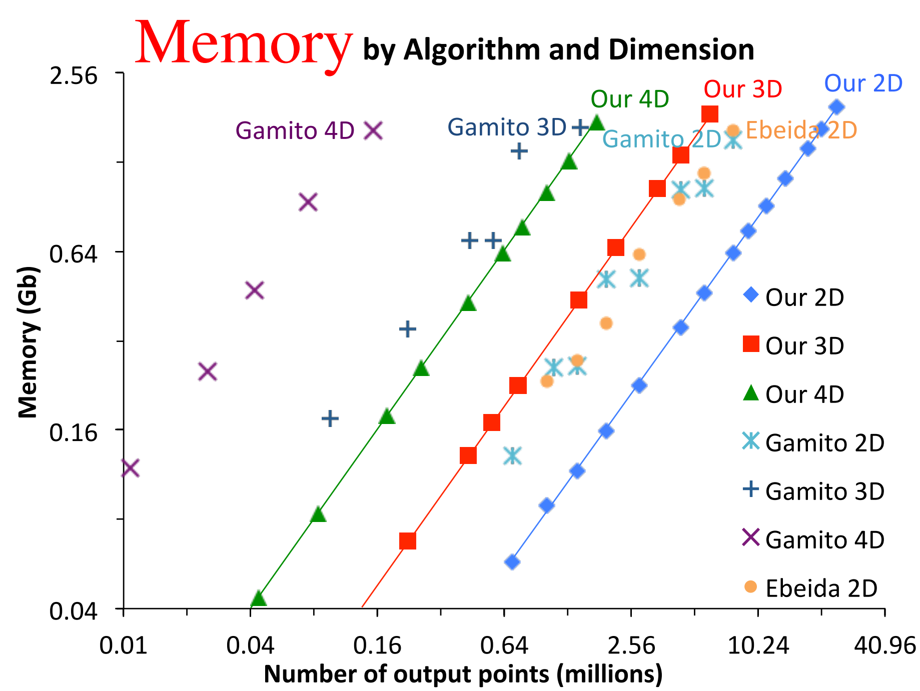
Time and Memory

vs. true quadtrees (Gamito), polygons (Ebeida 2D)

all linear in both, but constants matter

log-log scales





Memory savings from simpler datastructure

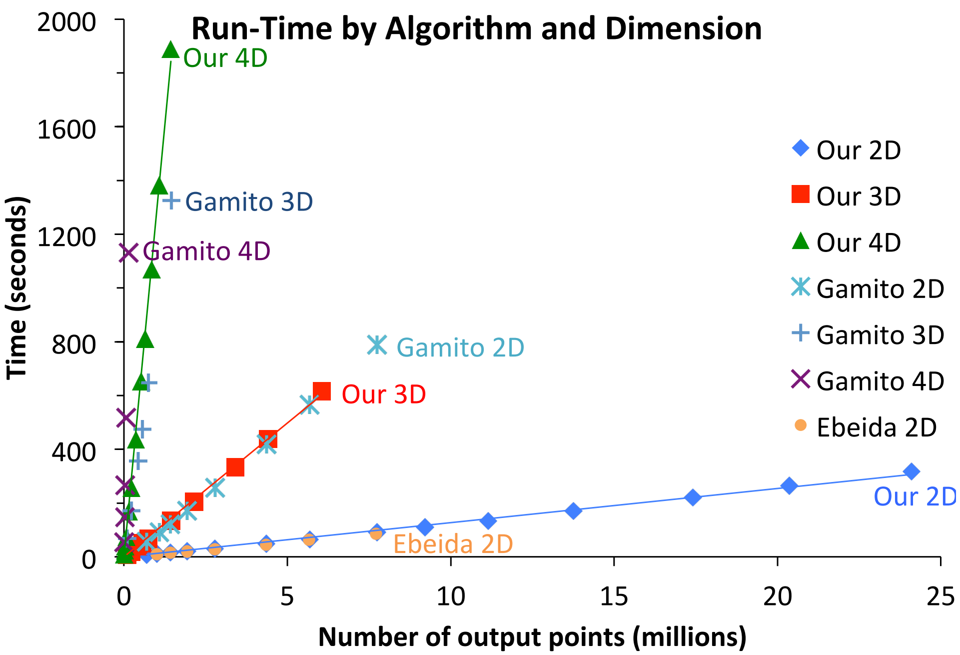
Time savings from that + simpler/fewer checks

Sandia National Laboratories

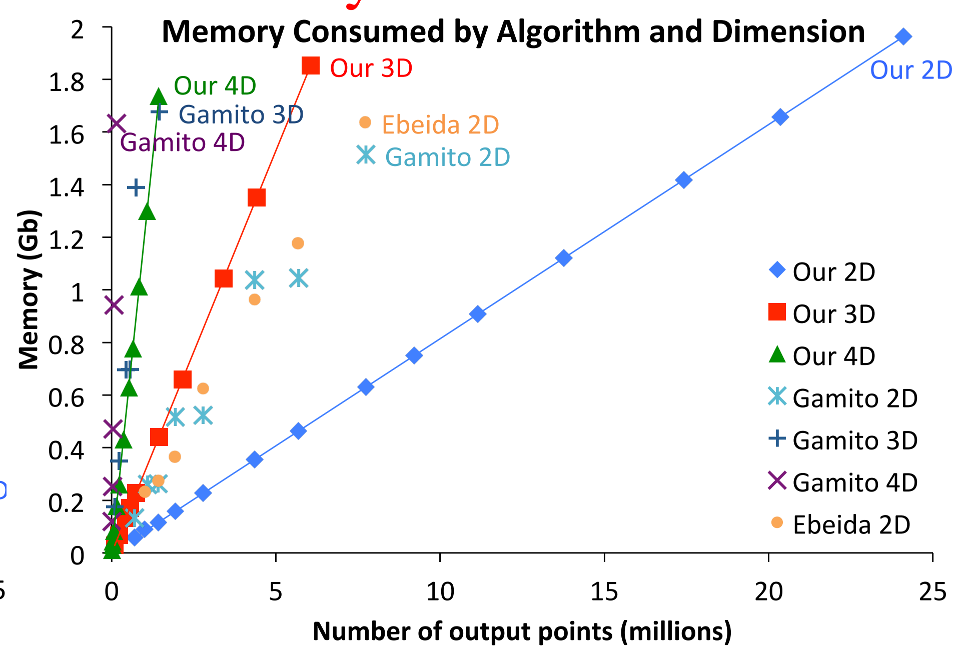
Time and Memory Theory

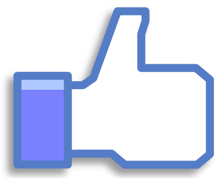
- **Run-time**
  - **Practice: linear in #points, grows by dimension**
  - **Proof: not available**
    - **Spatial statistics, expected area fraction of cells? And where?**
- **Memory**
  - **Linear in #points**
  - **No dynamic memory allocation**

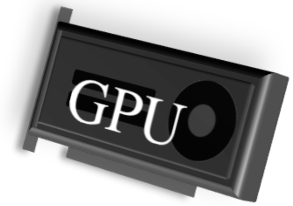
Time



Memory





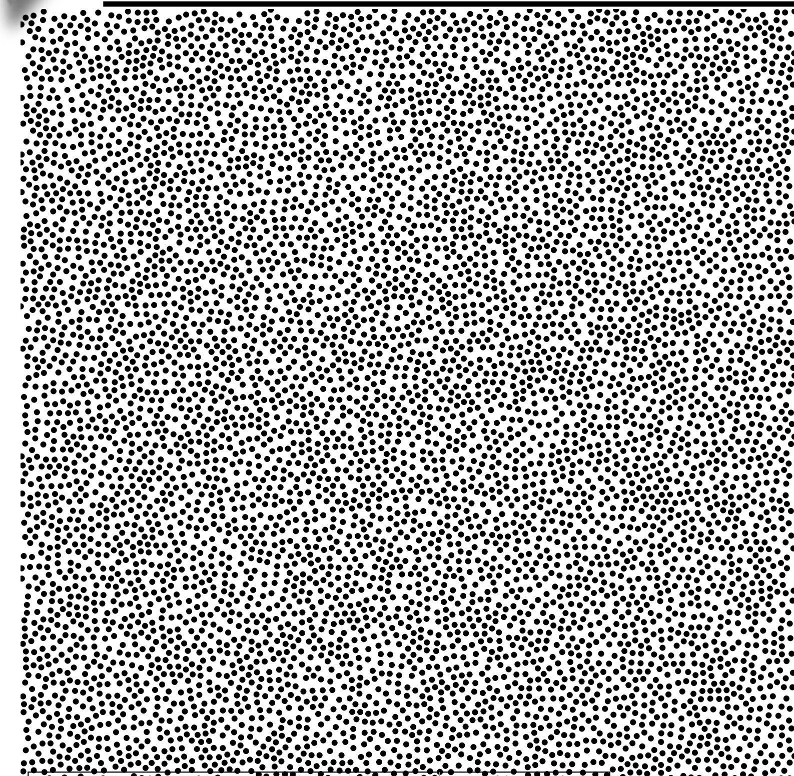


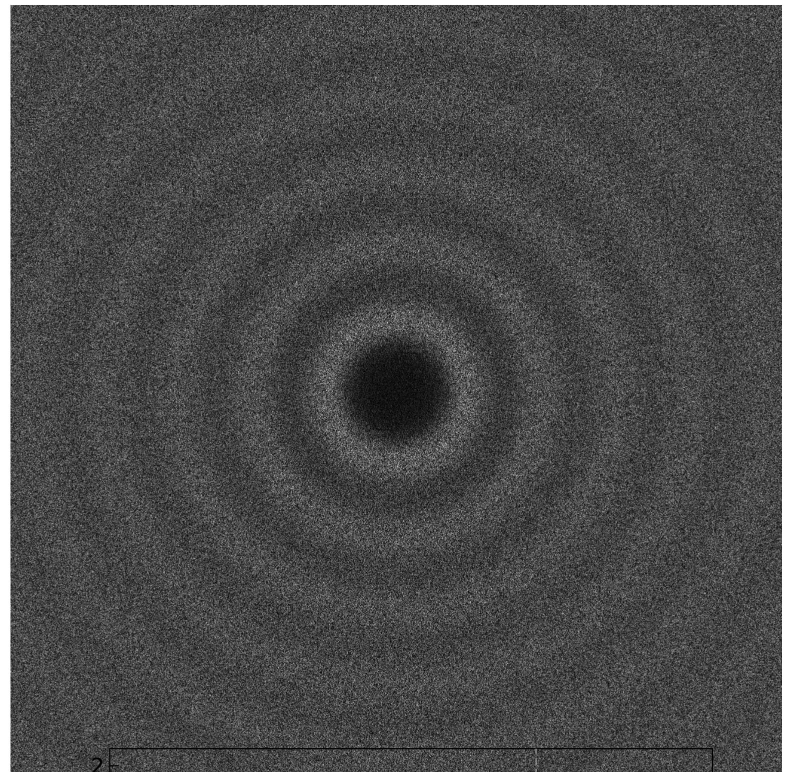
- Rejection sampling is great on a GPU
  - Nothing to communicate for a dart miss!

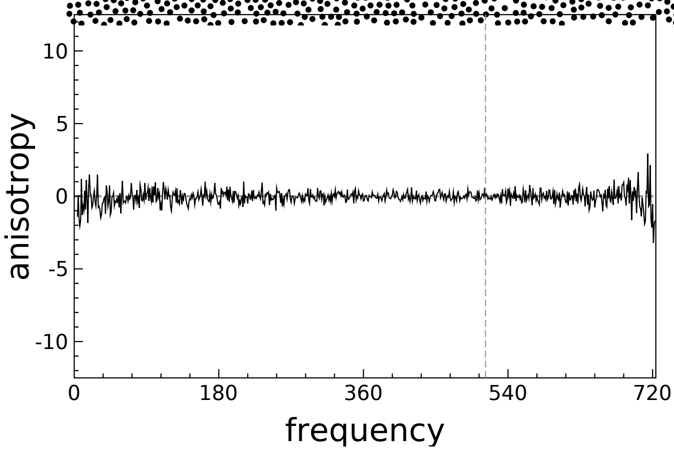
- 10x speedup on NVIDIA GTX 460
  - Memory-limited to 600k points 2d, 200k in 3d

Point Cloud Quality?

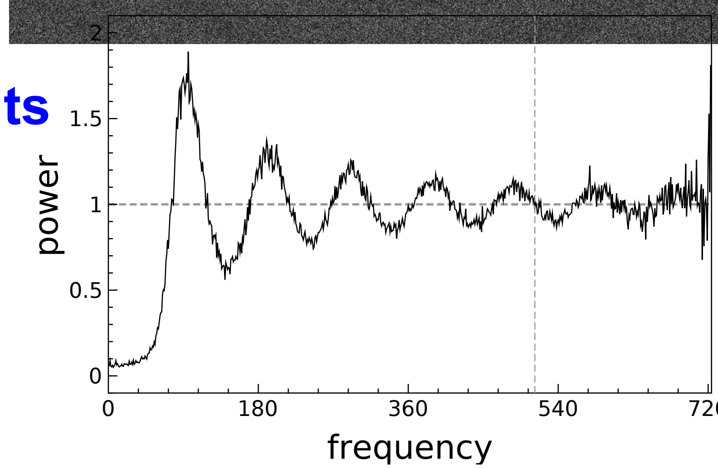
Provably correct bias-free, maximal up to precision



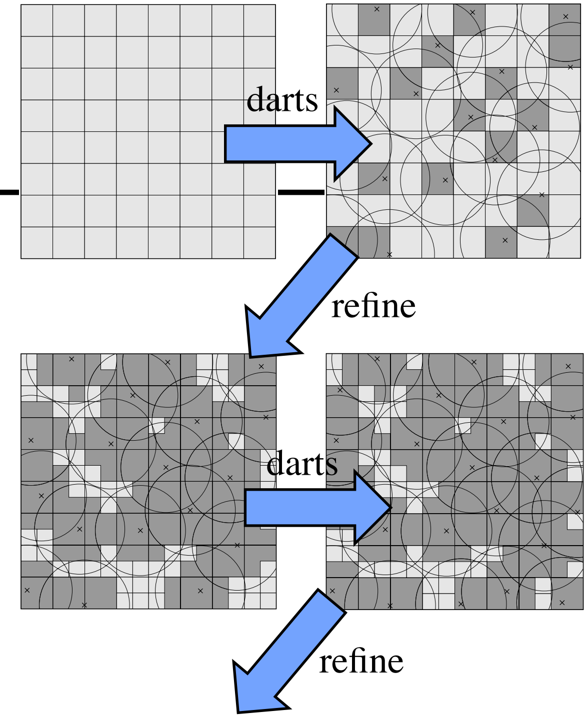




Experiments confirm (GPU)



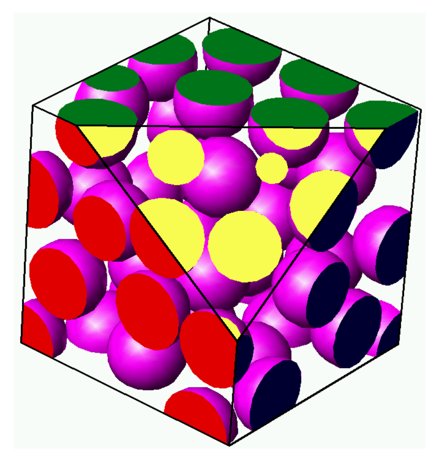
Conclusions



- MPS Maximal Poisson-disk sampling
  - Simpler, faster, less memory
  - Three simple ideas
    - Flat quadtree
    - Constant # throws / ignore misses
    - Global refinement
  - CPU and GPU

Reviewer #0: "The paper is yet another one about faster Poisson-sampling, but I see that it is significantly faster, uses less memory, is just simpler, easier to implement, and works well for higher dimensions."

- Future, dimensions > 4?
  - Not so great, quadtrees too big
- Two bonus thoughts…



UCDAVIS
UNIVERSITY OF CALIFORNIA

Sandia National Laboratories

Bonus thoughts

- Definition of desired result vs. process to obtain it (e.g. algorithm)
- Which would you rather have?

Bonus thoughts

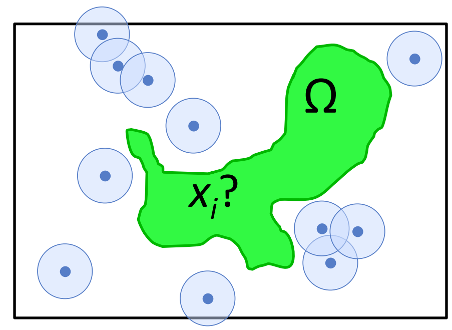
- Trick question!
- E.g. sorted order vs. bubblesort process
- Ax=b vs. Gaussian elimination

- A definition of desired output enables the discovery of new means to obtain it.

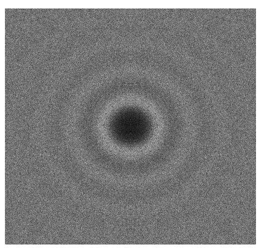
"Unbiased" Opinion

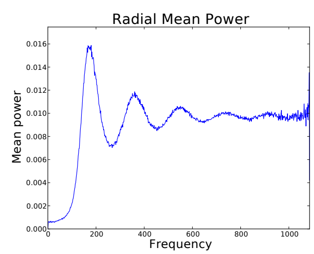
- **Unbiased as a description of (serial) process**
  - **insertion probability independent of location**

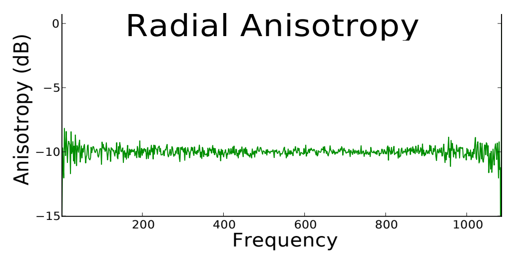
$$P(x_i \in \Omega) \propto \mathrm{Area}(\Omega)$$



- **Unbaised as a description of outcome**
  - **pairwise distance spectra, blue noise**







PSA code great for standard pictures

- **Unbiased process leads to unbiased outcome, but so might other processes**
  - **Opinion: need something beyond "viewgraph norm"**
  - **Need metrics for "how unbiased is it"**
    - **Define spectrum S that is the limit distribution of unbiased sampling, and standard deviations.**
    - **Our process generated S', and |S-S'| < 0.4 std dev (S)**

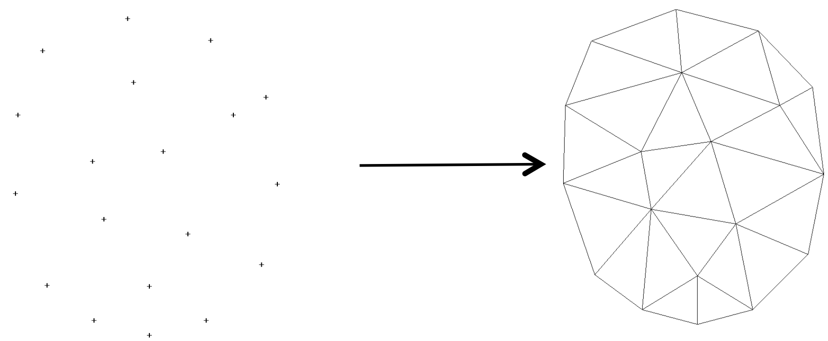
Sandia National Laboratories

Meshing and Triangulation Background

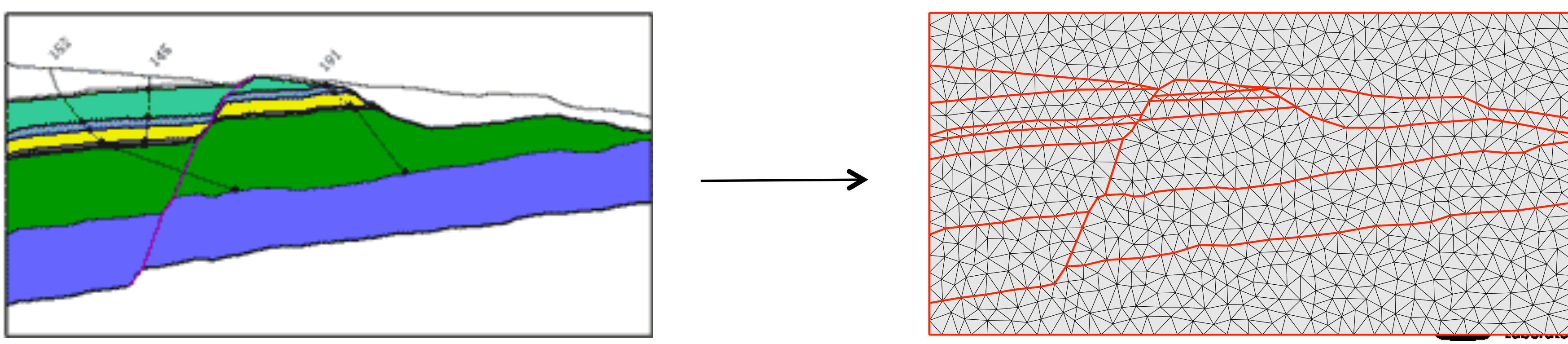
Connect those sample points!

Meshing and Triangulation

- Triangulating: point cloud -> triangles
  - Fixed input point positions

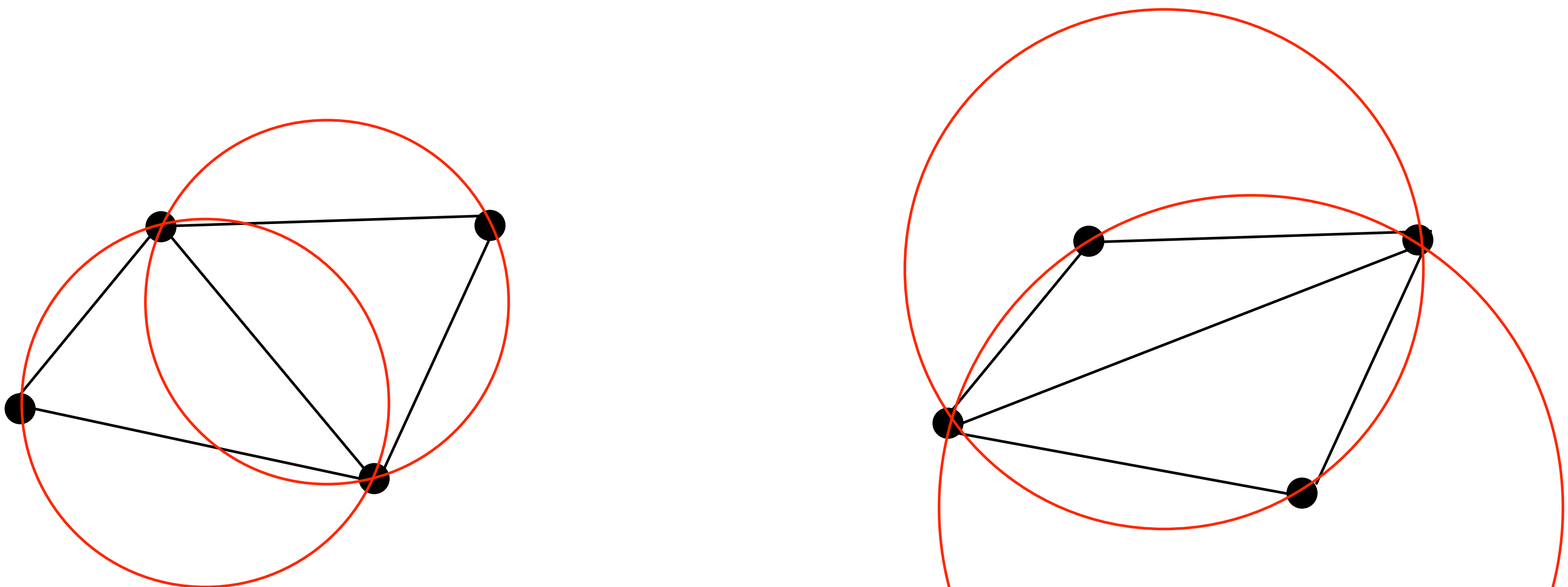


- Meshing: boundary representation -> points and triangles
  - freedom to put the points where you want
  - Subdivide input curves into edges, surfaces into triangles



Delaunay Triangulation

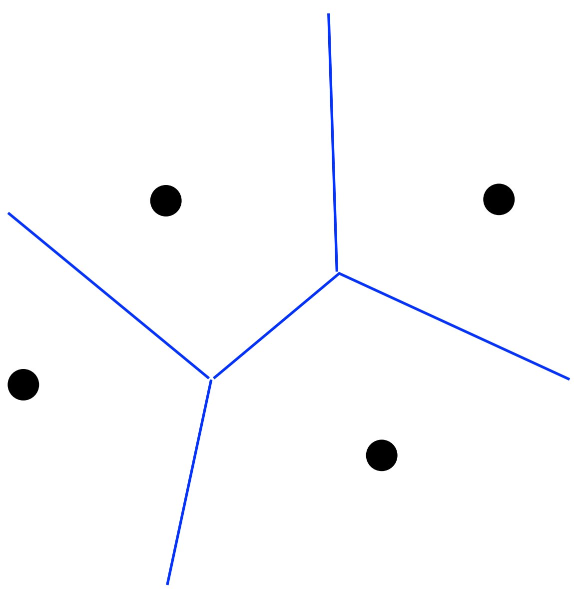
- Special role for both triangulating and meshing
- Given 4 points, two choices of diagonal edge

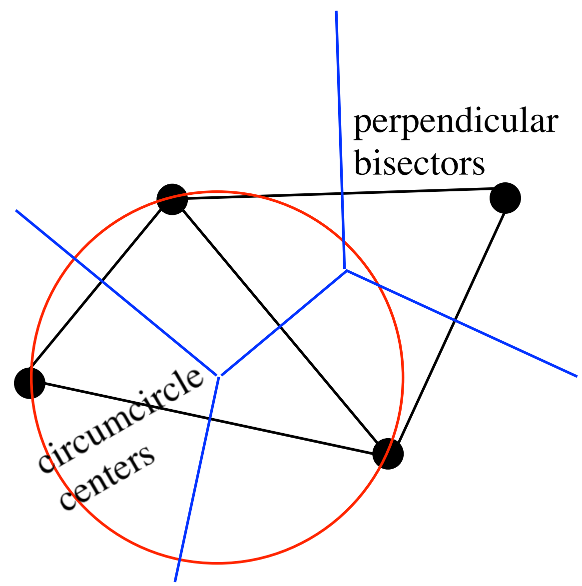


- Exactly one pair of circumcircles will be empty
- Maximizes minimum angle
- For more than four points, can check/flip locally to achieve global lexicographic max min angle

Voronoi Diagram

- **Region closer to that vertex than any other**





- **Voronoi vertices are (locally) furthest domain points from any black point**

Quick Quiz

- Which came first,
  - Delaunay Triangulation or Voronoi Diagram?

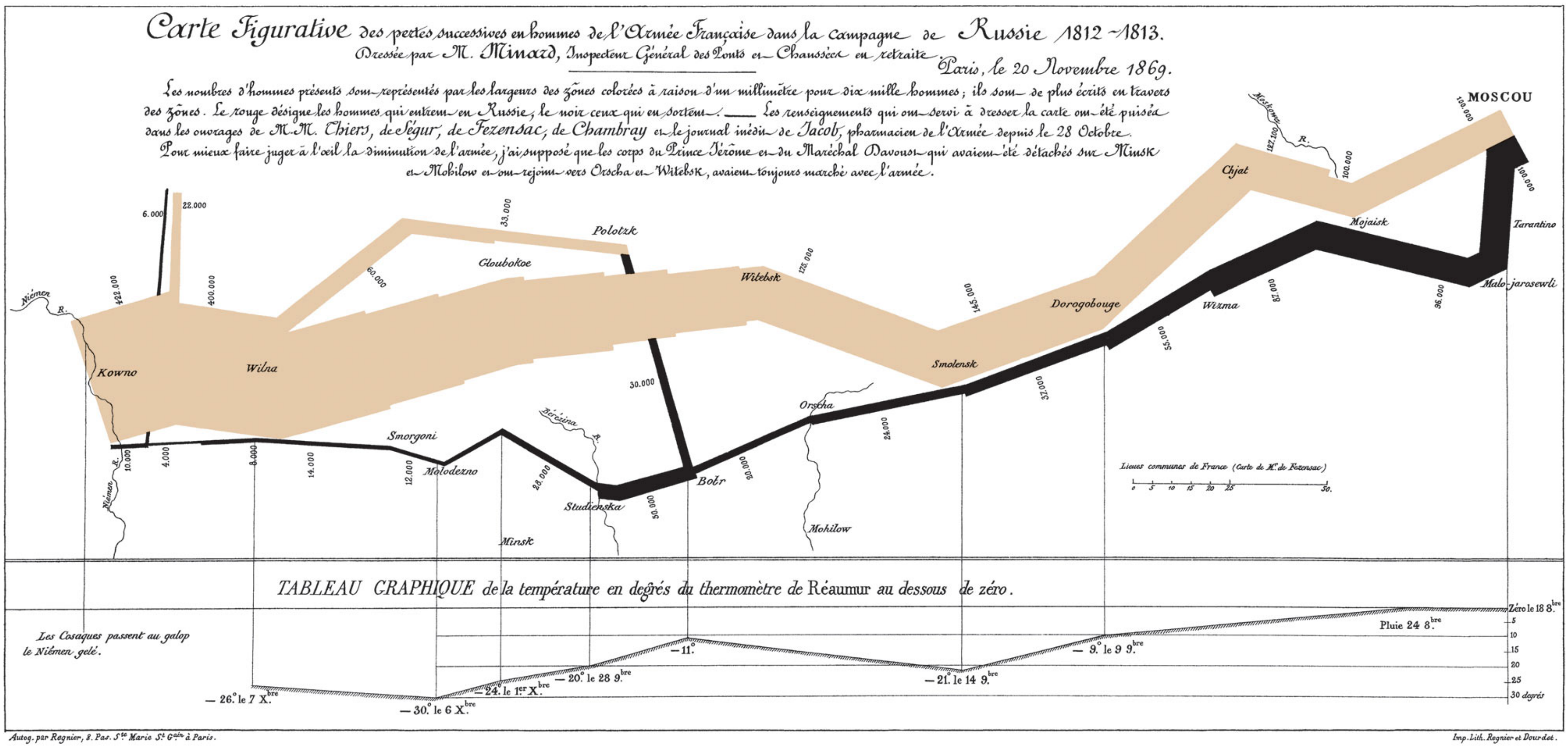
Voronoi Diagrams

- Descartes 1644
  - quadratic forms
- John Snow 1854
  - Broad Street pump, Soho, cholera
  - Data outlier
- Boris Delaunay 1934 paper

Quick Quiz

- **What does this, the most famous multidimensional display diagram in history, have to do with it?**

  Hint, how do you pronounce “Delaunay?”



Voronoi-Delaunay People

- Georgy Voronoy, 1908
- Boris Delaunay, 1934 paper, lived 1890-1980
- Both Russian citizens and published in French
  - Advisor and student, Delaunay named Voronoi diagrams after his advisor who worked on them
  - Mountain climber (top 3 in Russia)

Efficient and good Delaunay meshes from random points

M. S. Ebeida[1], S. A. Mitchell[1], Andrew A. Davidson[2], Anjul Patney[2], Patrick Knupp[1] and John D. Owens[2]

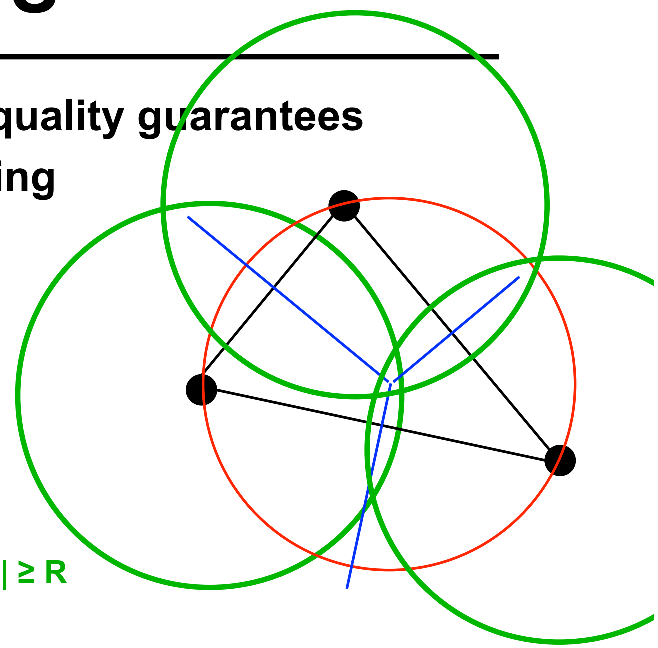
[1]Computing Research, Sandia National Laboratories

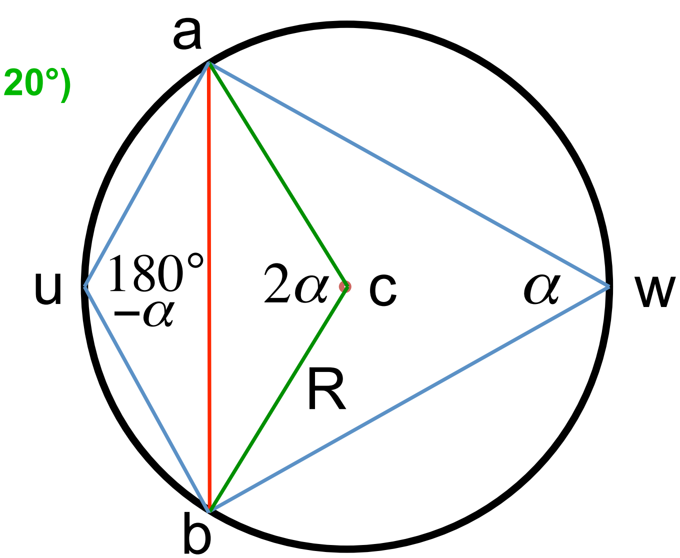
[2]Electrical and Computer Engineering, UCDavis

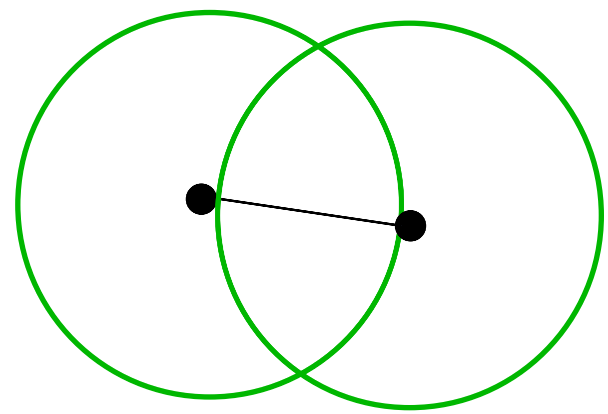
10/24/2011

Angles in DT of MPS

- **Random placement avoids structure, but plays no role in quality guarantees**
- **DT of maximal Poisson-disk sampling or any sphere packing**
- **Separated-yet-dense**
  - **Every domain point is covered by a disk, in particular every circumcenter**
    - **Circumcircle radius is at most the disk radius, recall circumcenter is a farthest point from a vertex**
    - **Longest edge is at most the circumcircle's diameter, $|e| \leq 2R$**
  - **No disk contains another point**
    - **Shortest edge (nearest neighbor distance) is at least disk radius, $|e| \geq R$**
  - **Central Angle theorem, ancient Greek**
    - **$\sin \alpha \geq |e| / 2R$**
  - **$\alpha \geq 30°$ (hence $\alpha \leq 120°$)**



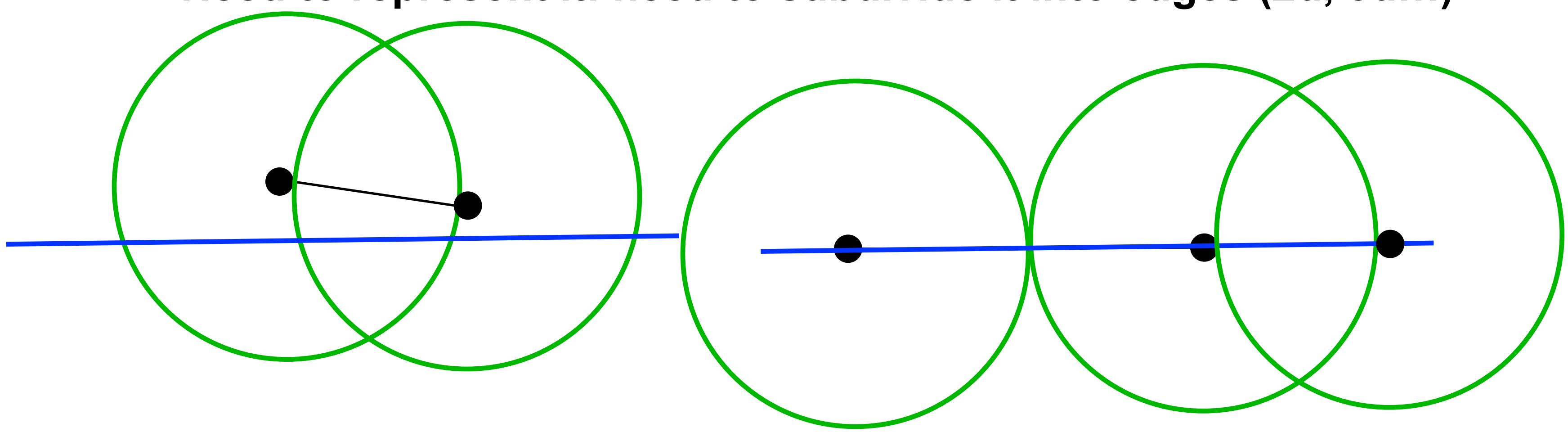




Sandia National Laboratories

Boundary Pre-Processing

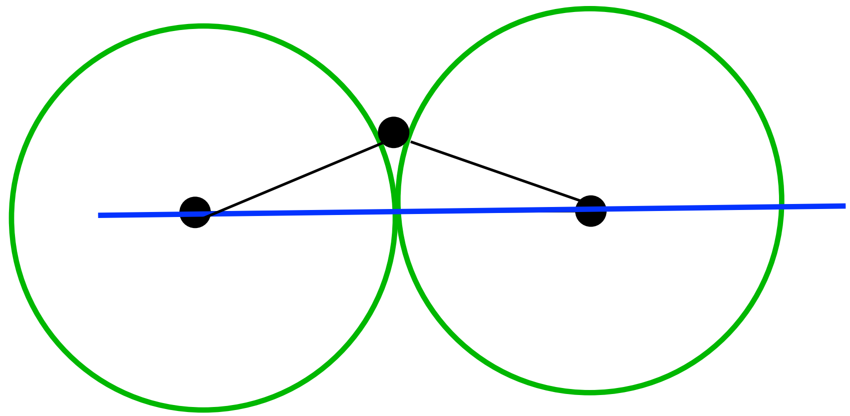
- Prior proof assumed entire plane covered by disks
- What about the domain boundary?
  - Need to represent it: need to subdivide it into edges (2d, 3d…)



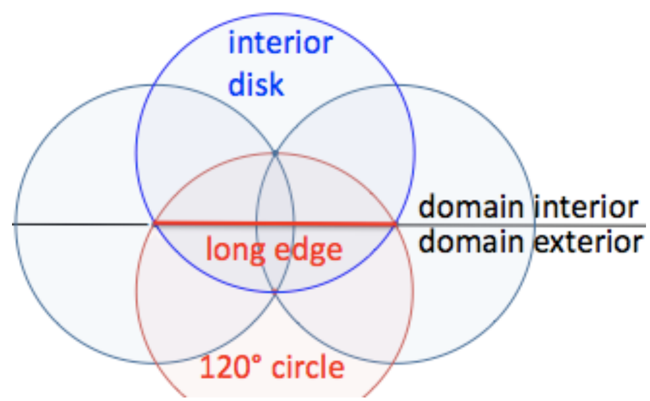
Randomly sample in 1 less dimension
Easy to get distances in [R,2R]

Boundary Sampling

- For edges in [sqrt(3) R, 2R], a sample on the surface could be too close, small & large angles.

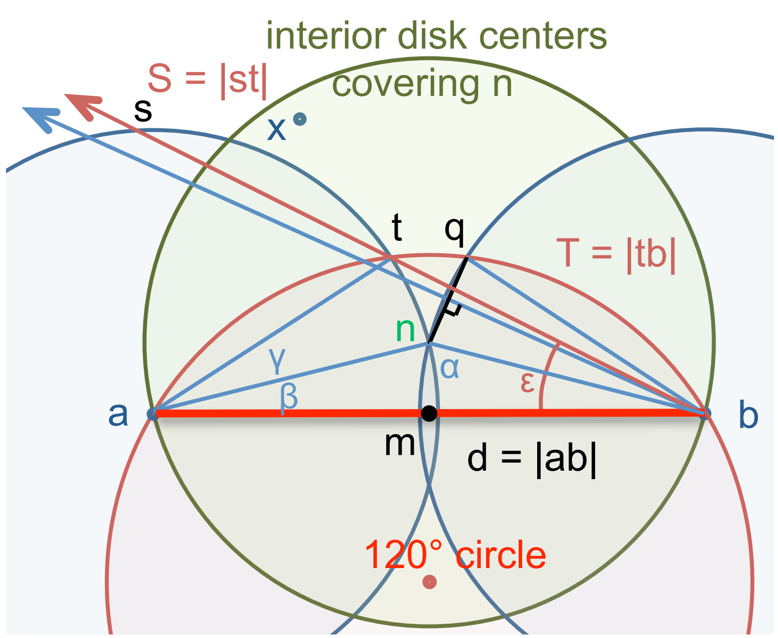


- Solution 1: sample boundary with R’ = sqrt(3)/2 R
  – Expand all disks to R before sampling interior.

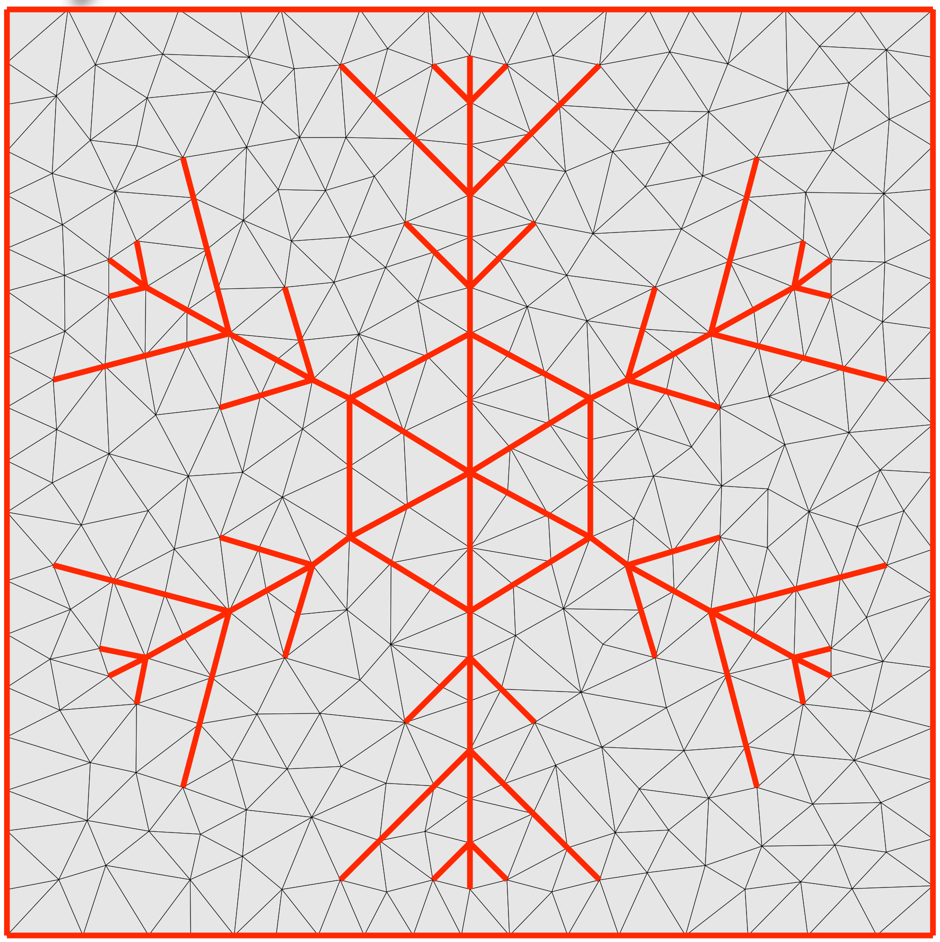


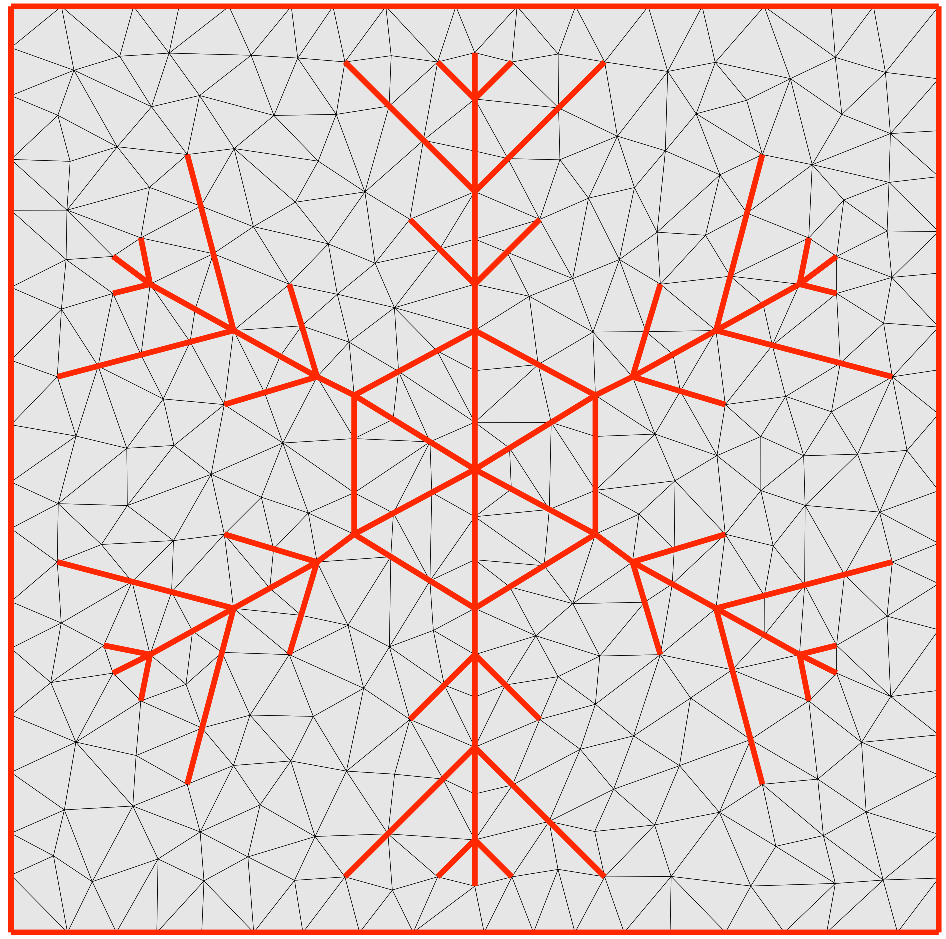
Boundary Sampling

- **Solution 2.**
  - **Sample bdy with R-disks.**
  - **Sample interior near bdy**
    - **Sample within R of boundary-circle intersection, outside 120° circle (it will cover all of ▽tqn)**
  - **Sample rest of interior.**

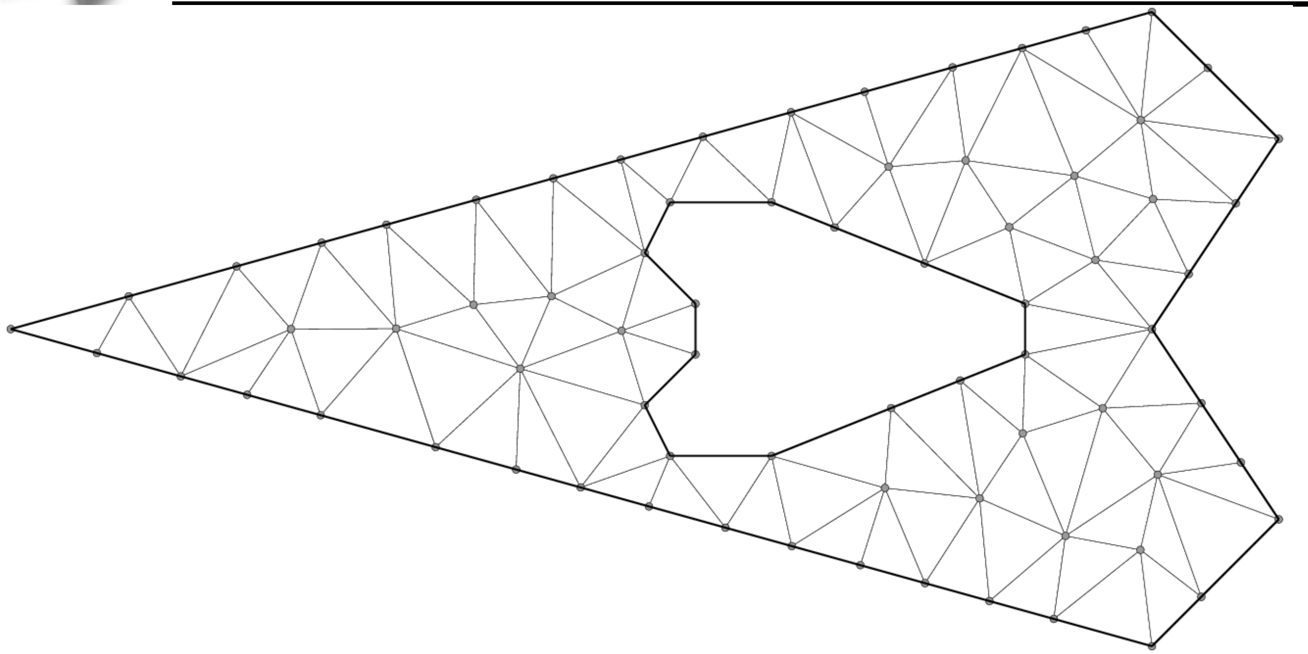


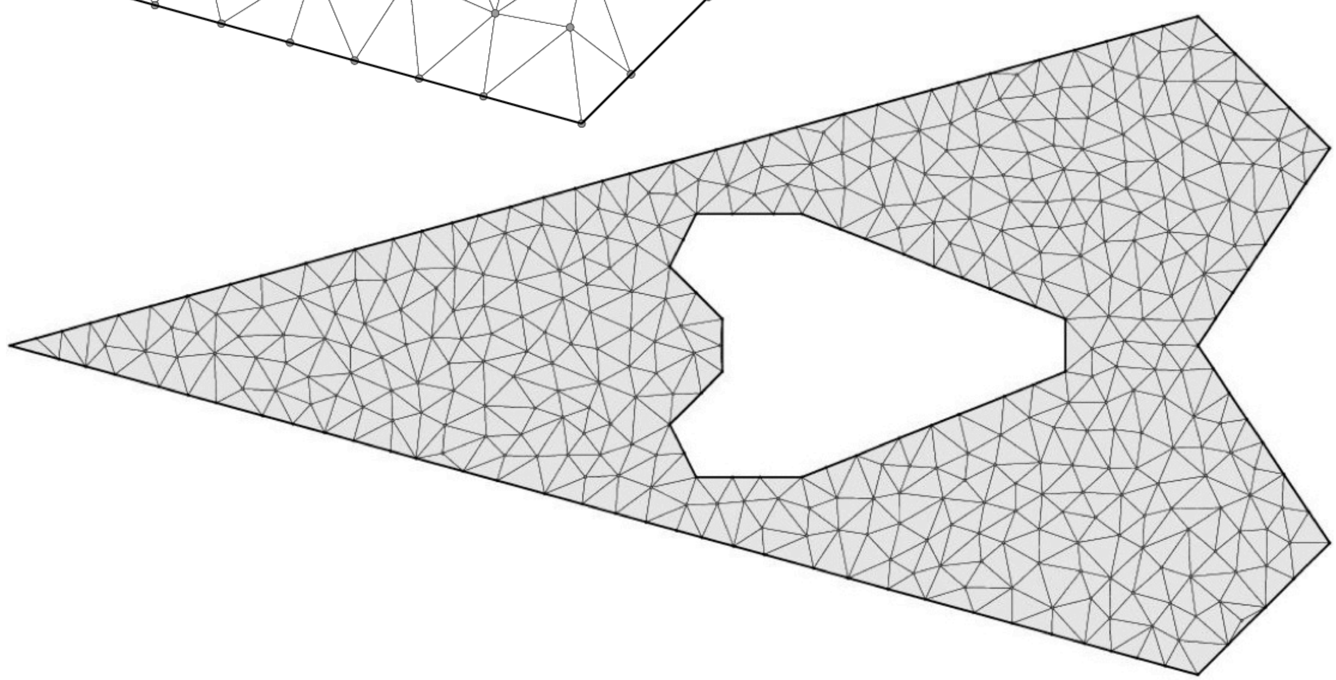
Example meshes





Example Meshes

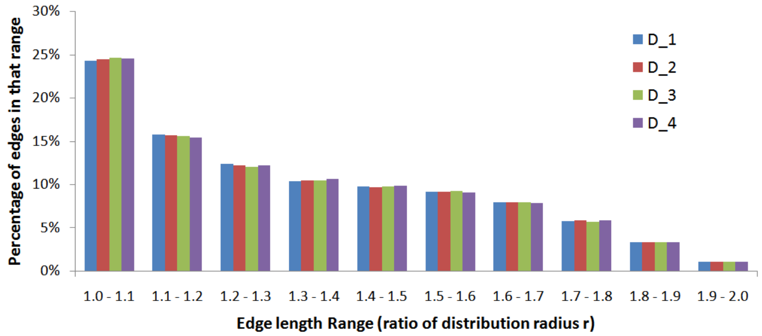


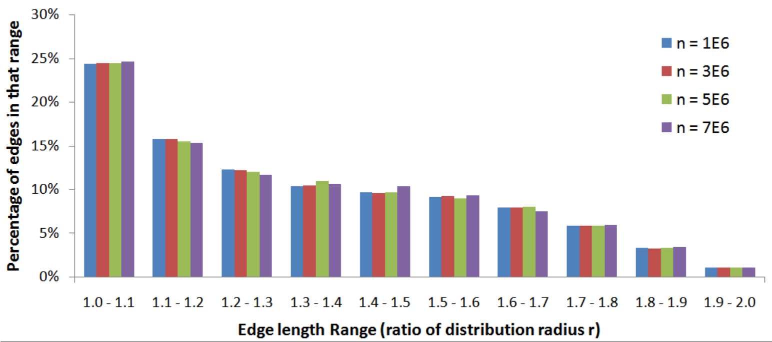


How does Maximal Poisson-disk sampling affect meshing algorithms?

Delaunay Edge length

- bounded between r and $2r$
- Connectivity can be retrieved locally
- Linear time complexity
- Easier parallel implementation
- Nice distribution almost independent of the domain / no. of points

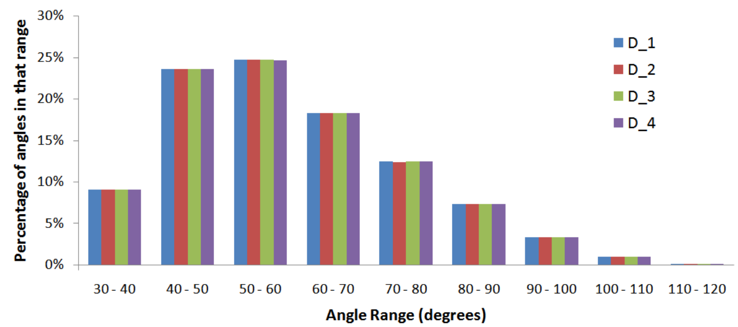


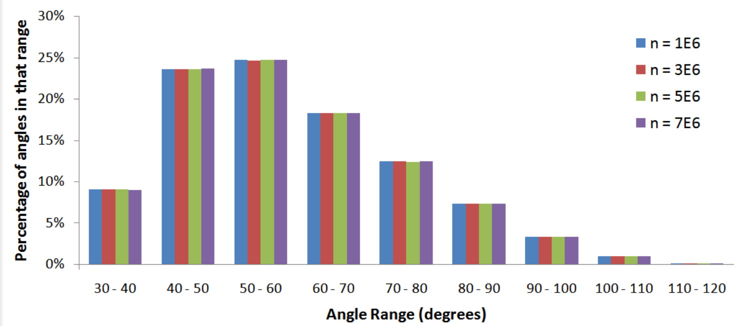


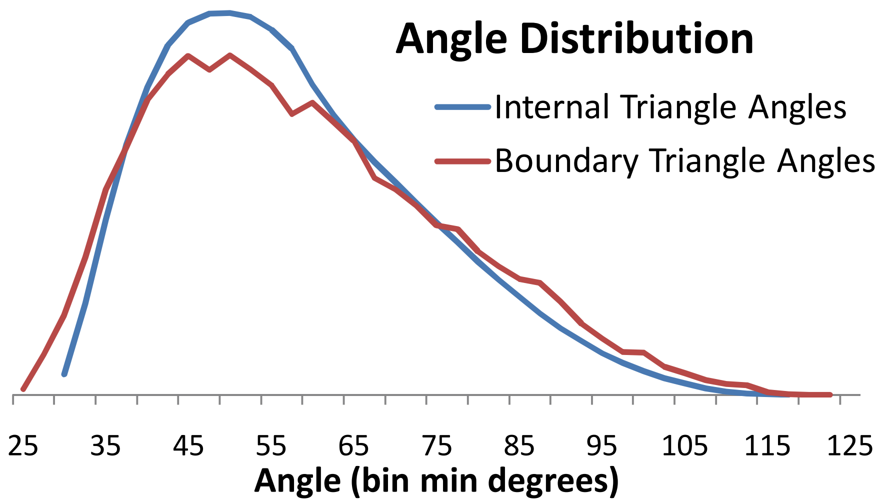
How does Maximal Poisson-disk sampling affect meshing algorithms?

Moreover

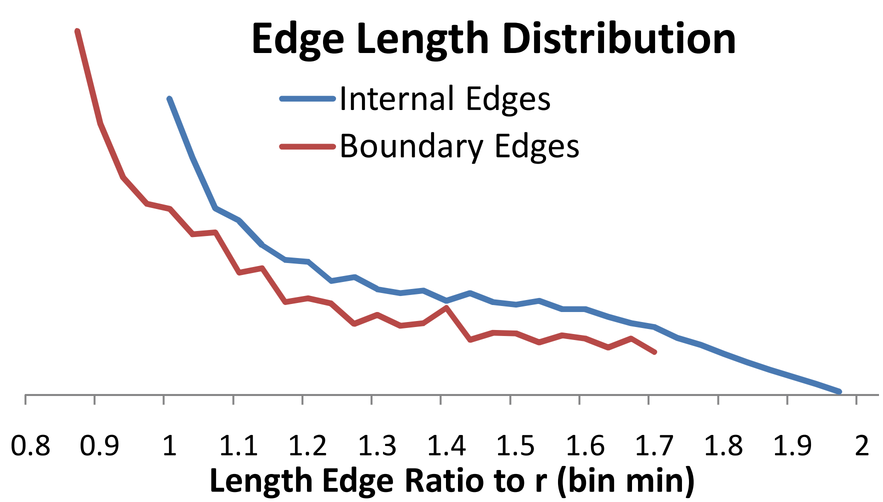
- Angles between $30°$ and $120°$
- Nice distribution almost independent of the domain / no. of points
- Easier handling of constrained input.
- Communication is only required in case of non-unique solutions.





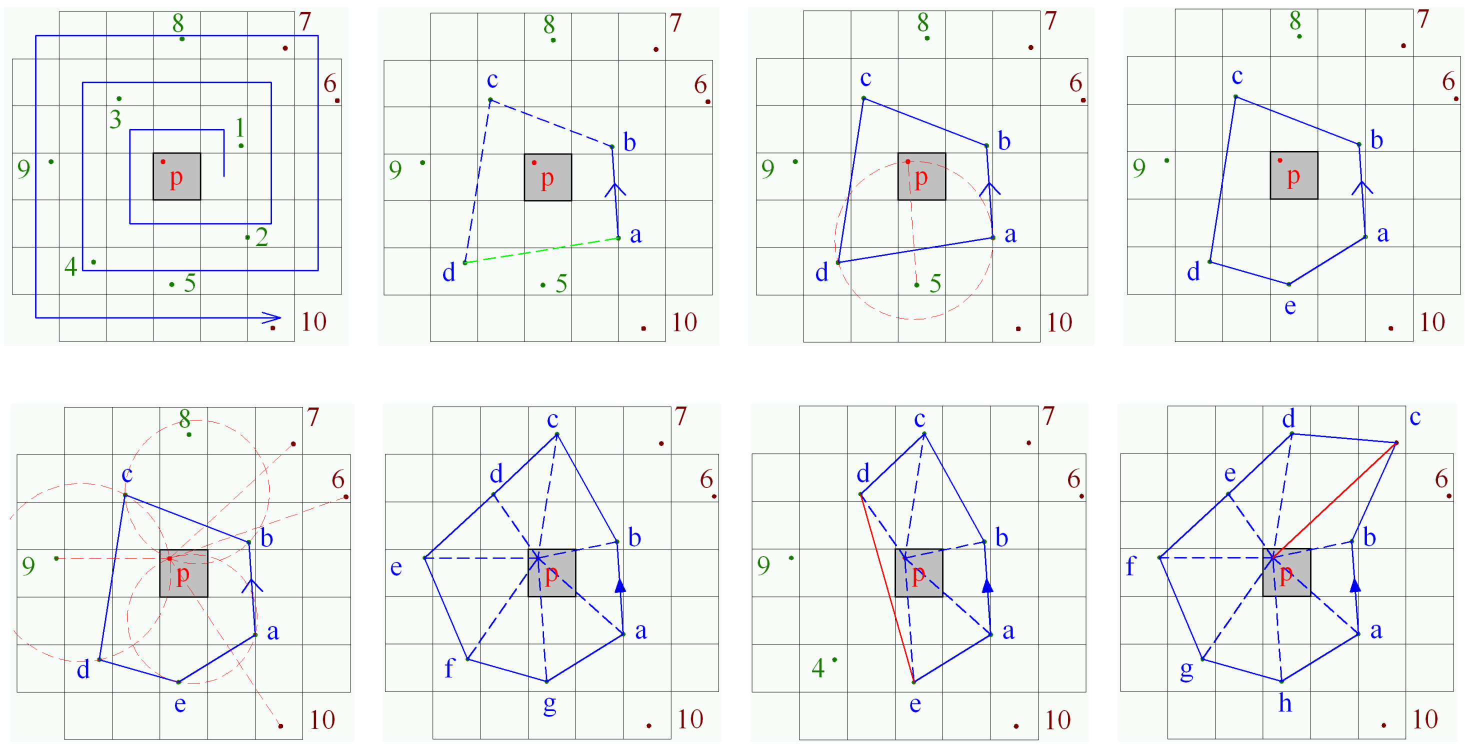


Angle Distribution
Internal Triangle Angles
Boundary Triangle Angles
25
35
45
55
65
75
85
95
105
115
125
Angle (bin min degrees)



Edge Length Distribution
Internal Edges
Boundary Edges
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
Length Edge Ratio to r (bin min)

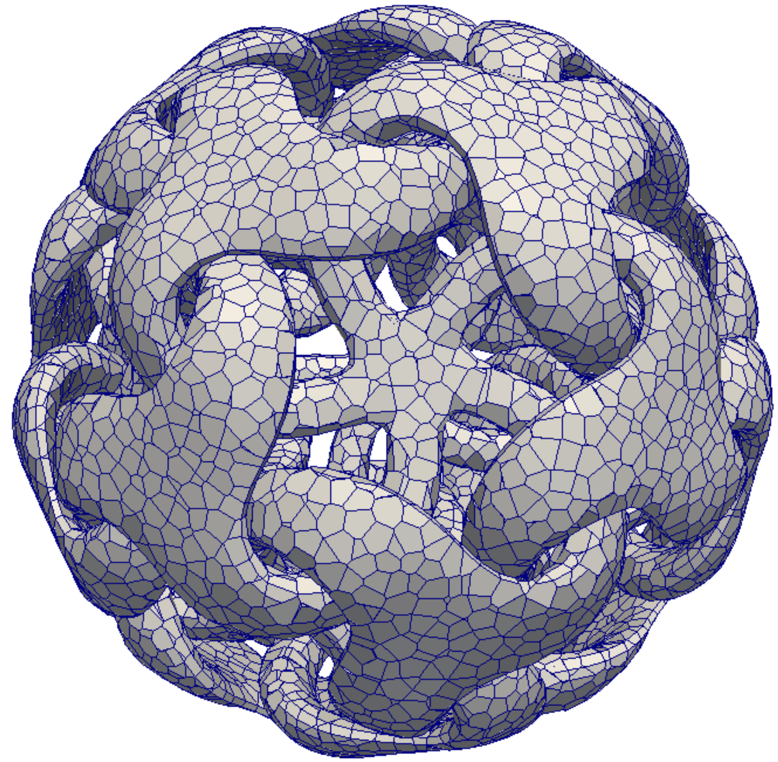
1. An Indirect method using a novel CDT algorithm (SIAM-GD 2011)



We were able to process 1 Million points in 2.7 seconds using a modern laptop.

Notre Dame de Paris, Gargoyle, by FreiGurita (1968)

+two other 2011 papers



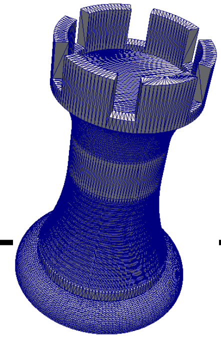
Uniform Random Voronoi Meshes

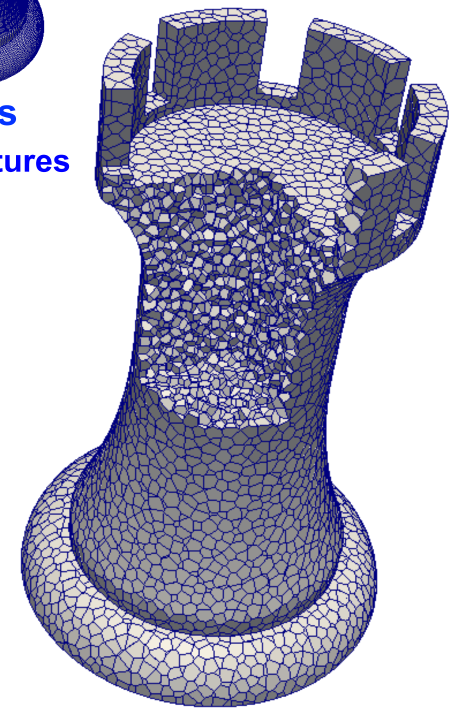
Mohamed S. Ebeida & **Scott A. Mitchell** (speaker)

20th International Meshing Roundtable
Paris, France

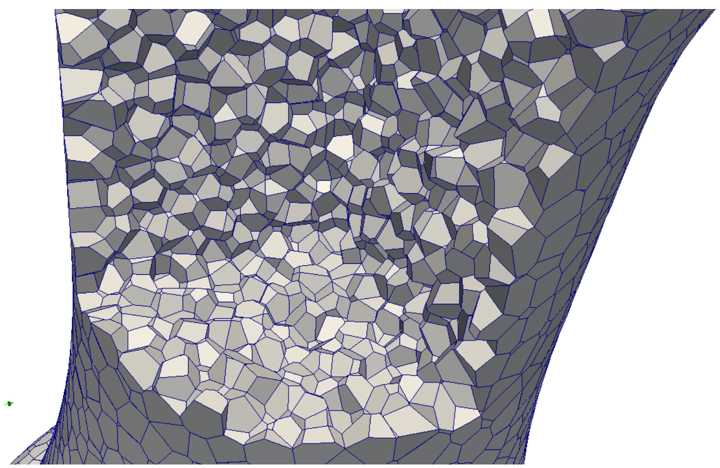
Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

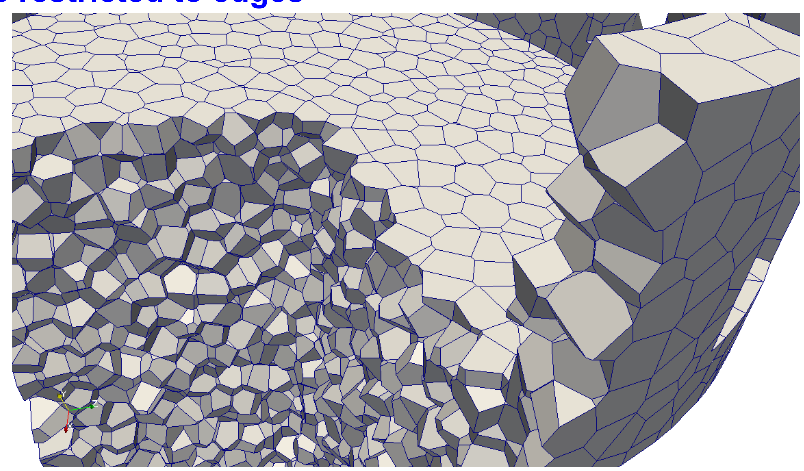
Summary





- Random Polyhedral Meshing
  - Generate random points using the maximal Poisson-disk process
    - Points placed on reflex boundary features, but not concave or flat features
    - Contrast to primal methods
  - Symbolically split points (not in paper)
  - Construct Voronoi cells
    - Bounding box, cut by boundary and Voronoi planes
      - Bounding box works because cells have bounded size
    - Small edges collapsed
- Get
  - Voronoi mesh of convex polyhedral cells
  - Bounded cell aspect ratio and facet dihedrals
  - Random orientation of mesh edges
    - Needed for fracture mechanics where cracks are restricted to edges

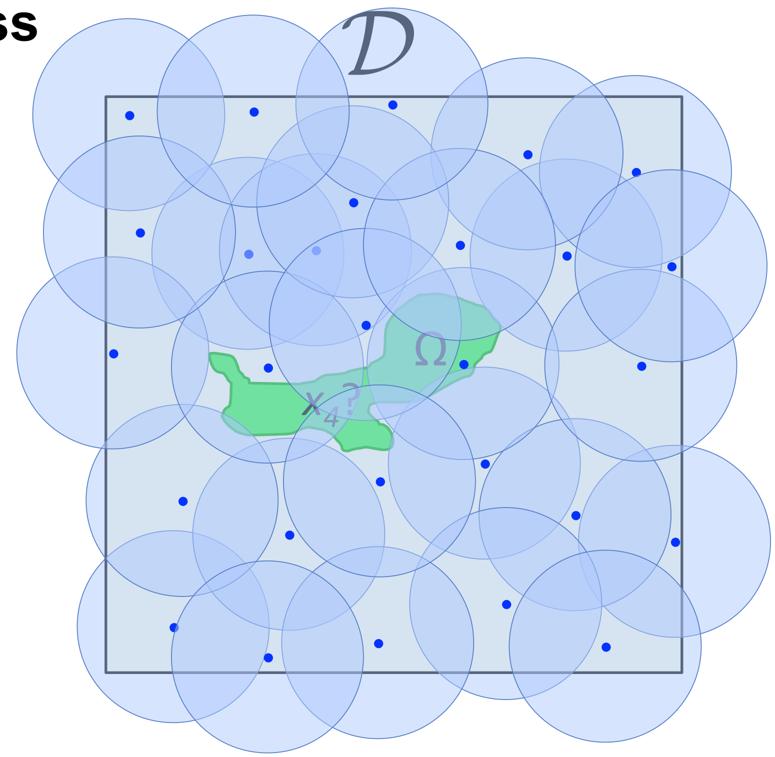




Maximal Poisson-Disk Sampling (MPS)

- **What is MPS?**

  - **Dart-throwing**
  - **Insert random points into a domain, build set X**
    - **With the “Poisson” process**



Empty disk: $\forall x_i, x_j \in X, x_i \neq x_j : ||x_i - x_j|| \geq r$

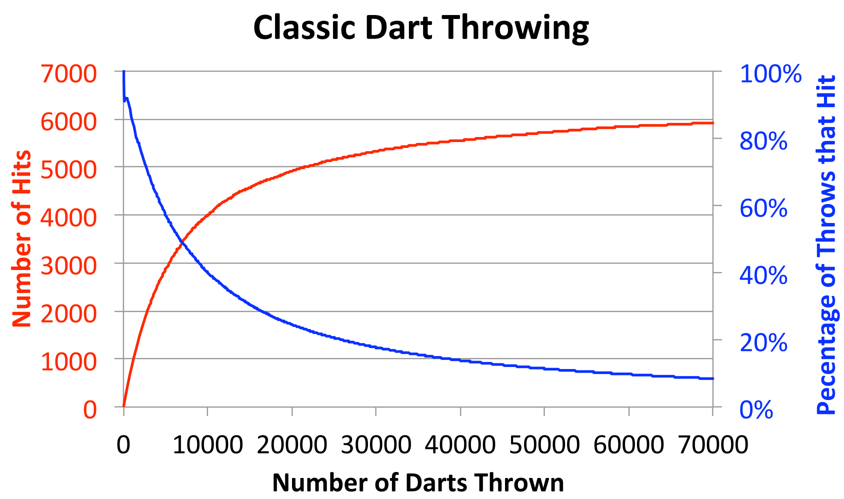
Bias-free: $\forall x_i \in X, \forall \Omega \subset \mathcal{D}_{i-1} :$

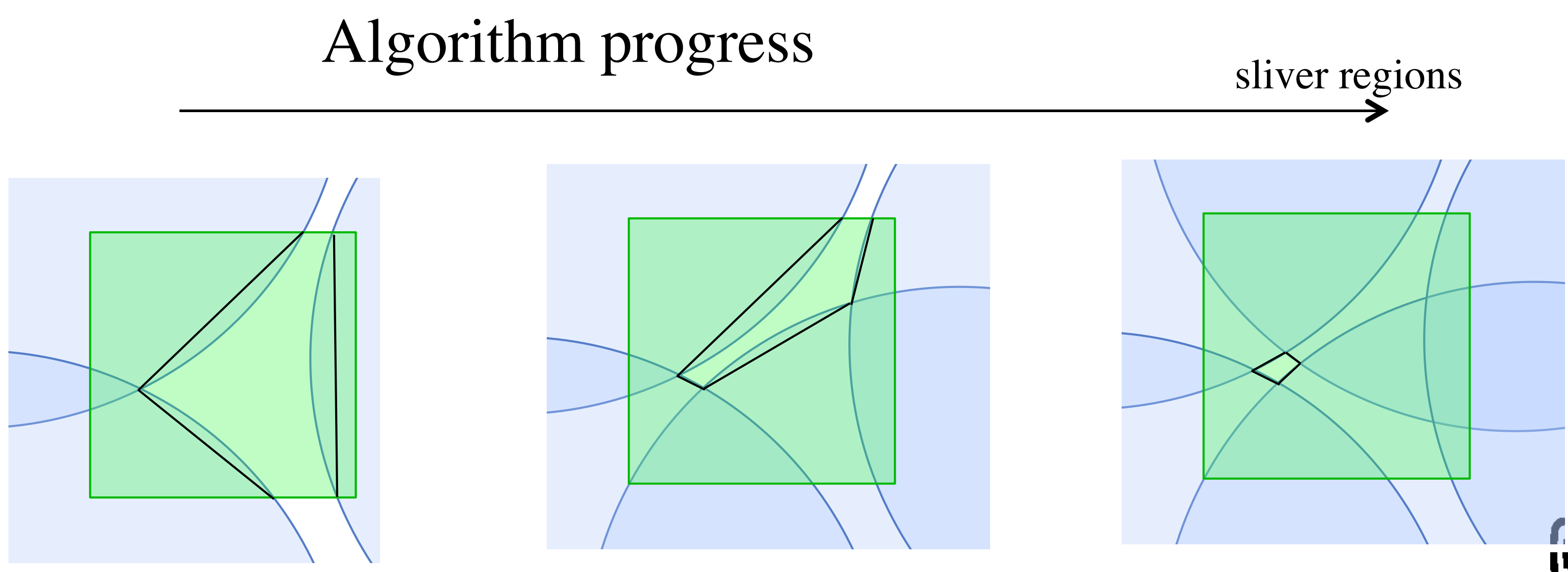
$$P\left(x_i \in \Omega\right) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$$

Maximal: $\forall x \in \mathcal{D}, \exists x_i \in X : ||x - x_i|| < r$

Sandia National Laboratories

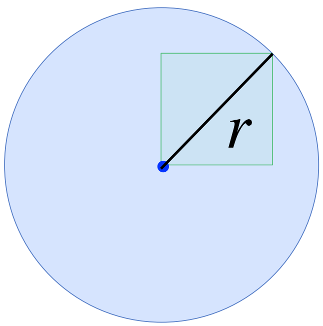
Statistical Process ≠ Algorithm



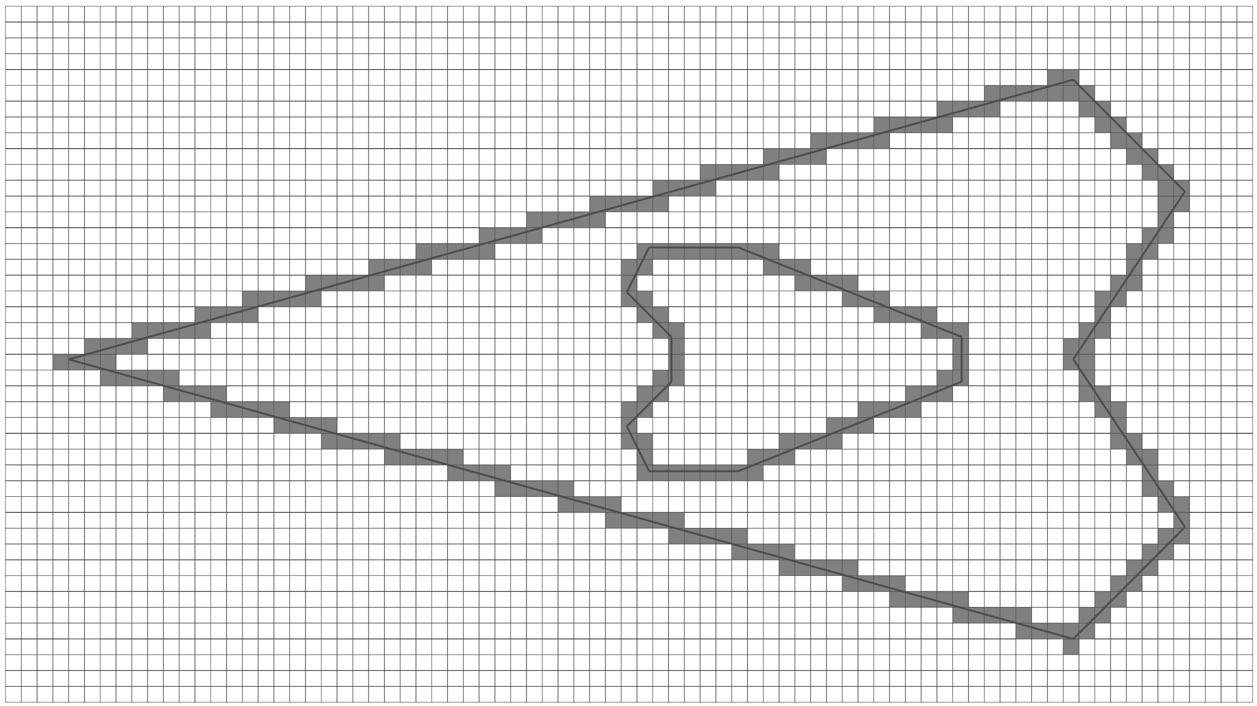


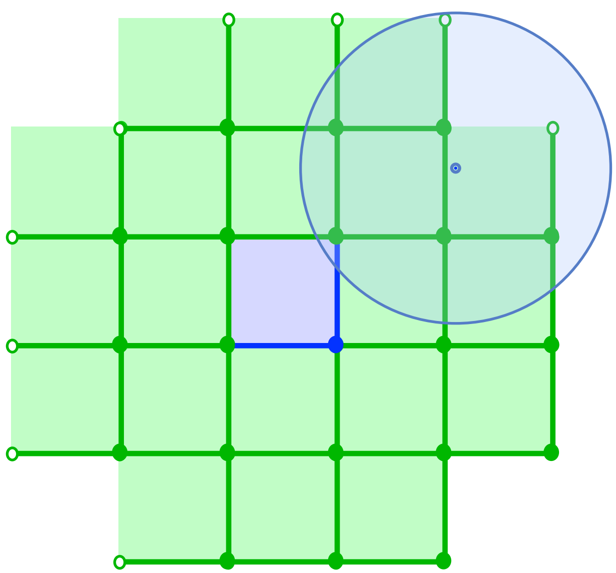
“Efficient maximal Poisson-disk sampling”

First provably correct, time- space-optimal algorithm.
Mohamed S. Ebeida, Anjul Patney, Scott A. Mitchell,
Andrew Davidson, Patrick M. Knupp, and John D. Owens.
ACM Transactions on Graphics (Proc. SIGGRAPH 2011), 30(4), 2011.



Background grid of squares (cubes…) for locality

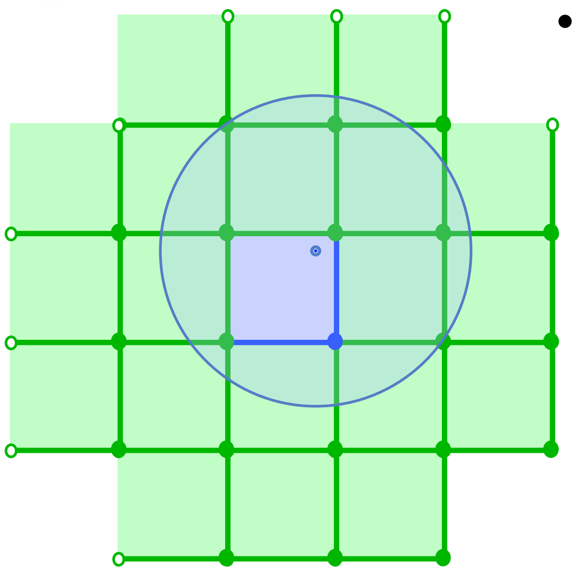




Sid Meier’s Civilization Template

Everything is O(1)

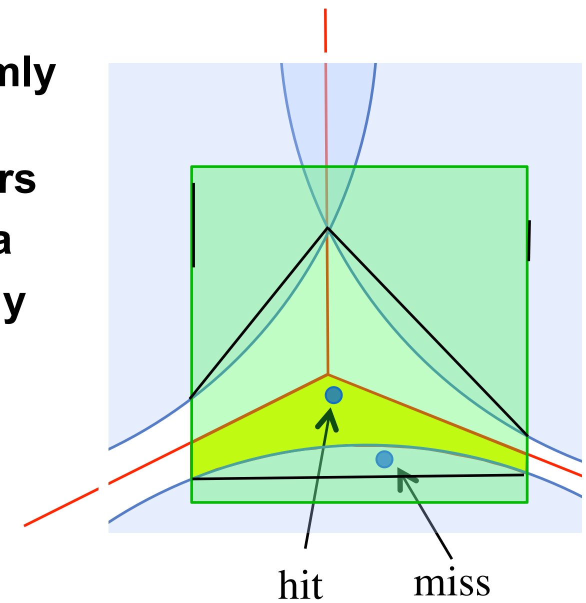
Efficient maximal Poisson-disk sampling

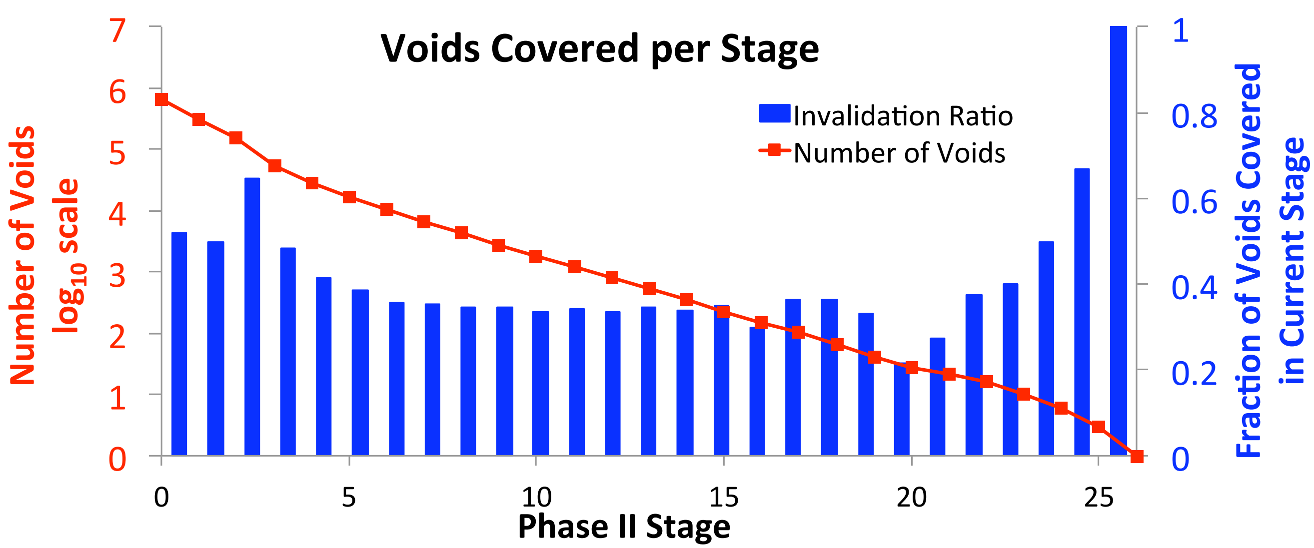


- **Algorithm**
  - **Phase I**
    Throw darts in squares
    - **Pick square uniformly**
    - **Pick point in square uniformly**
  - **Phase II**
    Throw darts in polygons $\supset$ slivers
    - **Pick sliver weighted by area**
    - **Pick point in sliver uniformly**

Bias-free: $\forall x_i \in X, \forall \Omega \subset \mathcal{D}_{i-1}$:
without selecting from entire domain

$$P\left(x_i \in \Omega\right) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$$





E(n) throws proof idea
- **Hit/miss ratio = Voronoi cell area ratio > constant.**

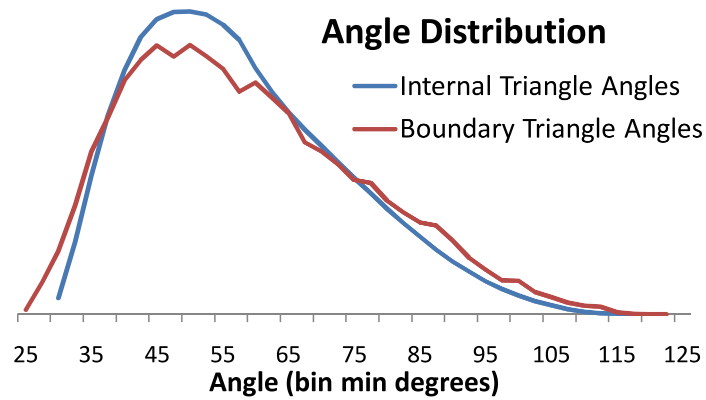
In practice, use flat implicit octree in d>2

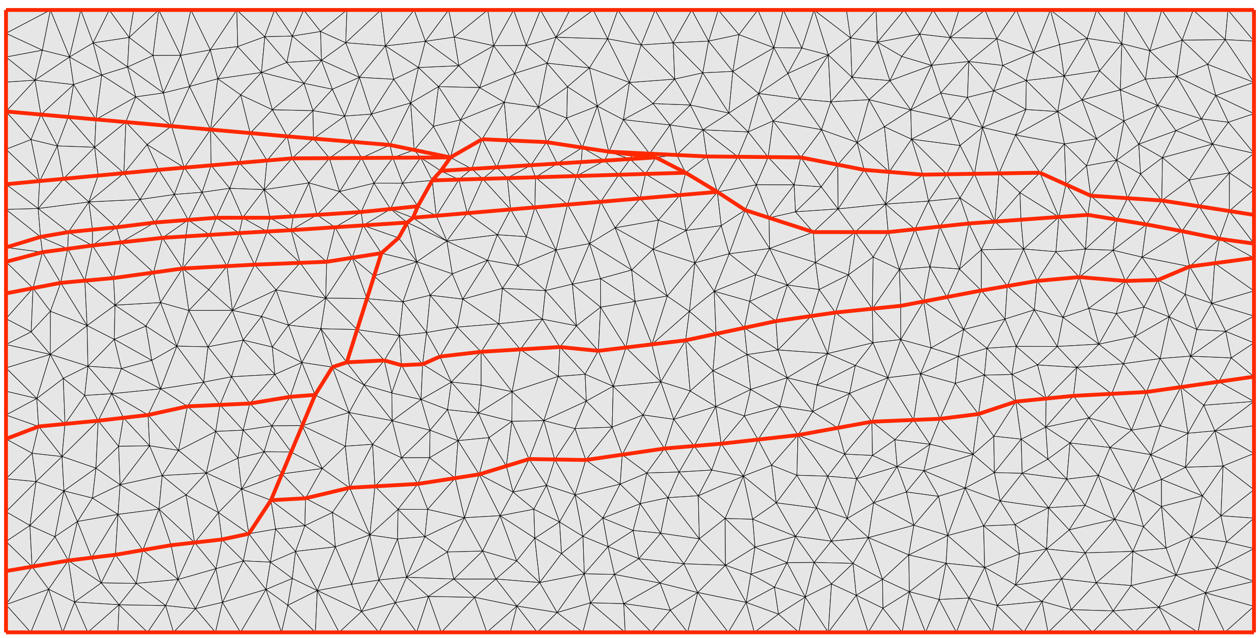
Sandia National Laboratories

Also Triangular Meshes

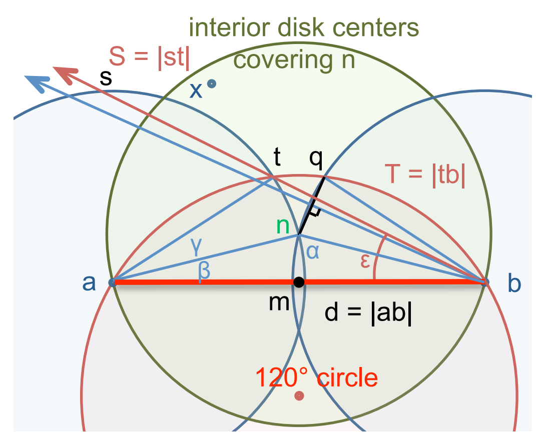
"Efficient and good Delaunay meshes from random points."
Mohamed S. Ebeida, Scott A. Mitchell, Andrew A. Davidson, Anjul Patney, Patrick M. Knupp, and John D. Owens.
Computer-Aided Design, 2011. Proc. 2011 SIAM Conference on Geometric and Physical Modeling (GD/SPM11).

- **Reverse cause-effect**
  - **Delaunay Refinement:**
    Insert circle-centers to kill large Delaunay circles
    - **Maximal sample results**
  - **MPS: Insert points randomly to maximally sample**
    - **Small Delaunay circles result**
  - **Nearly identical angle bounds either way**
    - **Delaunay circle-centers can be ignored!**





Cover the boundary with random disks

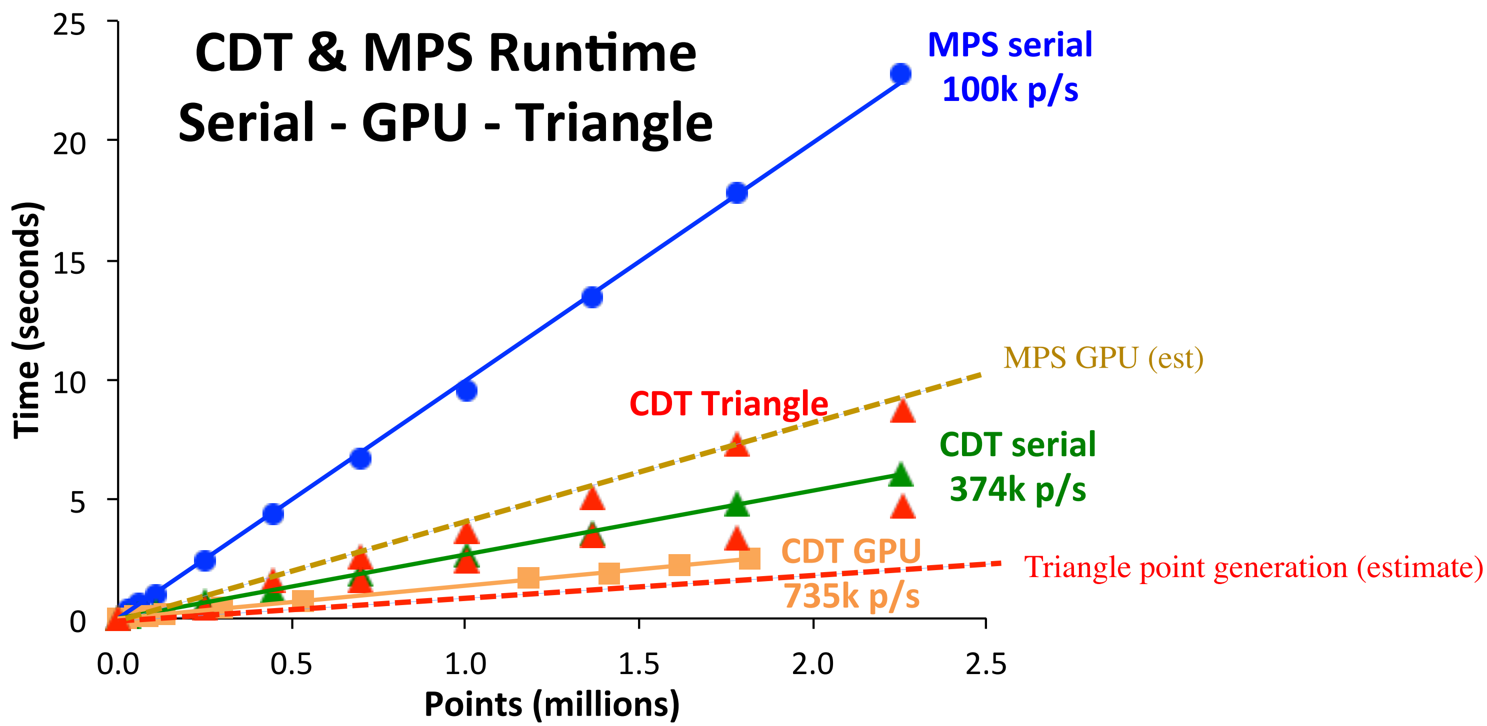


- **Simple algorithm for covering the boundary randomly**
  - **Complicated geometric proof**

Sandia National Laboratories

"Efficient" for MPS, scales great, but how fast?

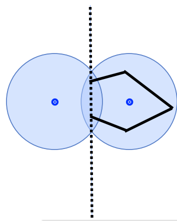
- Delaunay refinement
  - Points from deterministic process - fast
- MPS
  - Points from strict unbiased random process – slow
- But once points are generated we're as fast as Triangle, and our GPU code is 2x faster

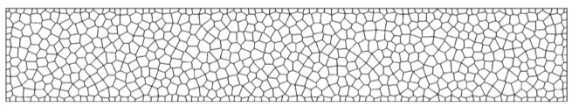


Sandia National Laboratories

What is MPS good for?

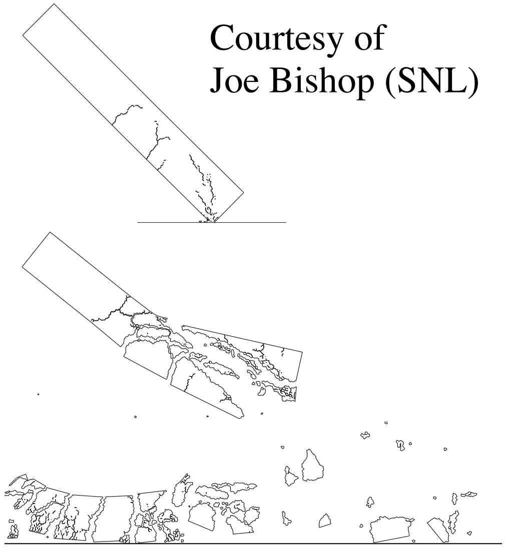
- Fracture mechanics simulations
  - Fractures occur on Voronoi cell boundaries
    - Mesh variation ⊂ material strength variation
      - Ensembles of simulations
    - Unbiased sampling gives realistic cracks
      - Edge orientations are uniform random
  - Domains: non-convex, internal boundaries



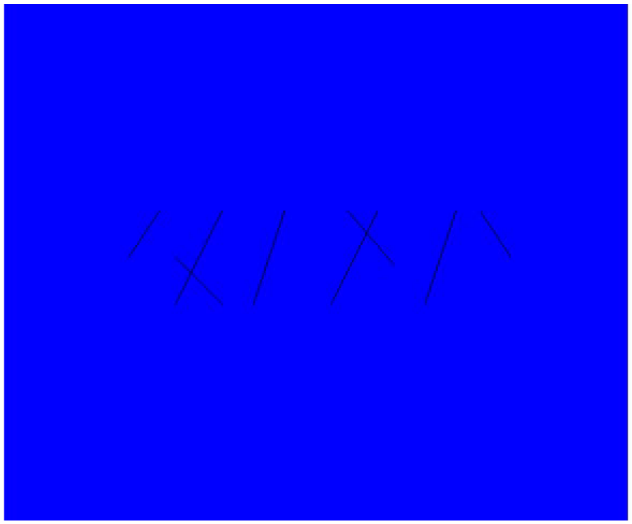


Fracture Simulations

Courtesy of
Joe Bishop (SNL)

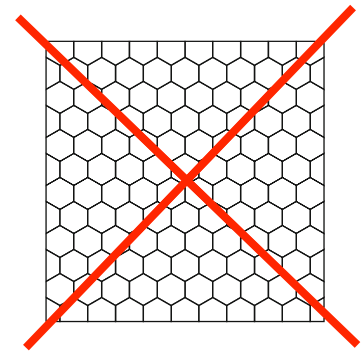


Impact
Joe Bishop, SNL org 1500
Fracture simulation
Need random meshes because
cracks are along edges



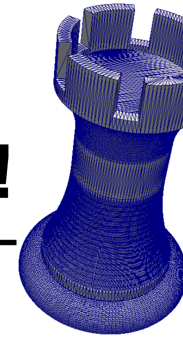
Sandia National Laboratories

Alternatives

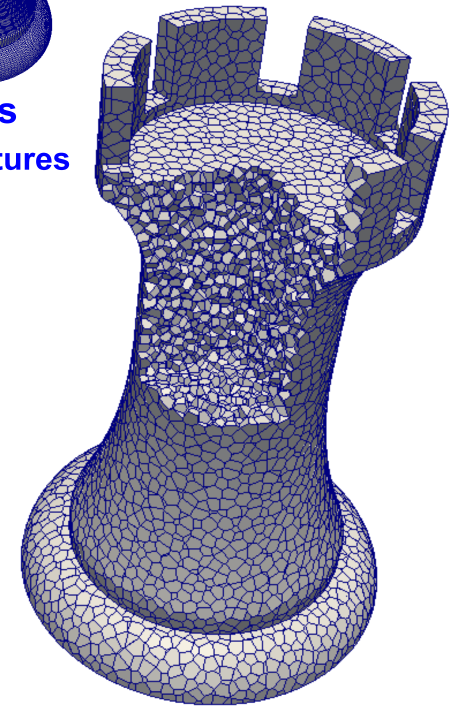


- Voronoi Mesher
  - CVT Centroidal Voronoi Tessellation
    - Seed = cell's center of mass
    - Via iterative adjustment of seed location
    - Good shaped cells, but "biased", regular mesh
    - Target app: fracture simulations with fracture along mesh edges
- Primal meshers
  - Miller: maximal disk packings for bounded edge-radius tet meshes
  - Shimada and Gossard Bubble meshes
    - Force network, insertion and removal

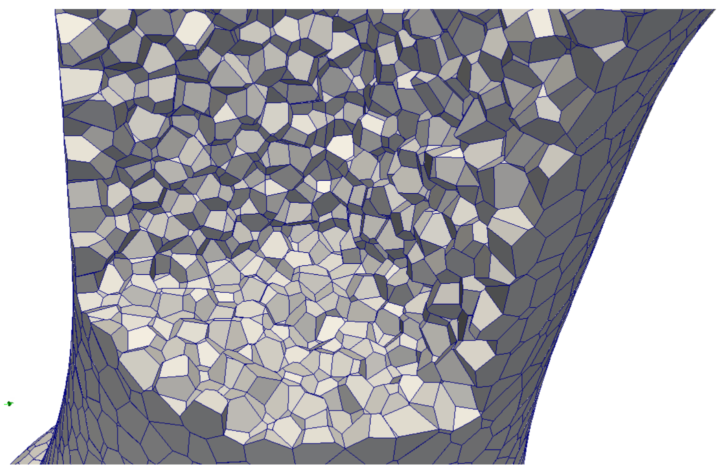
IMR paper algorithm!

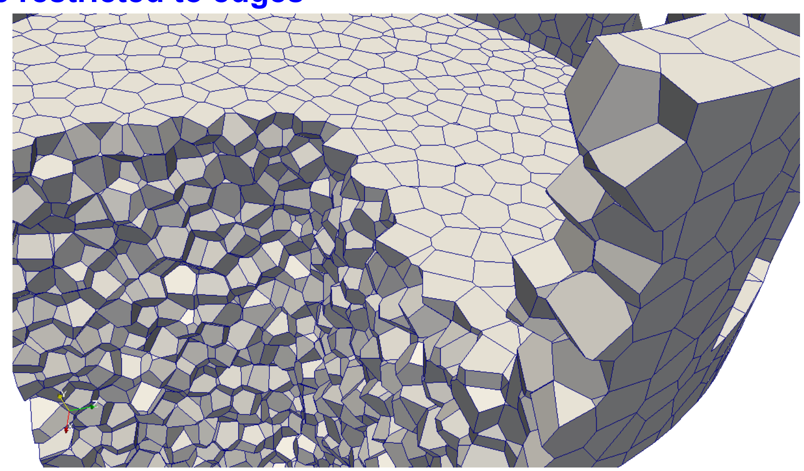


- Random Polyhedral Meshing
  - Generate random points using the maximal Poisson-disk process
    - Points placed on reflex boundary features, but not concave or flat features
    - Contrast to primal methods
  - Symbolically split points (not in paper)
  - Construct Voronoi cells
    - Bounding box, cut by boundary and Voronoi planes
      - Bounding box works because cells have bounded size
    - Small edges collapsed



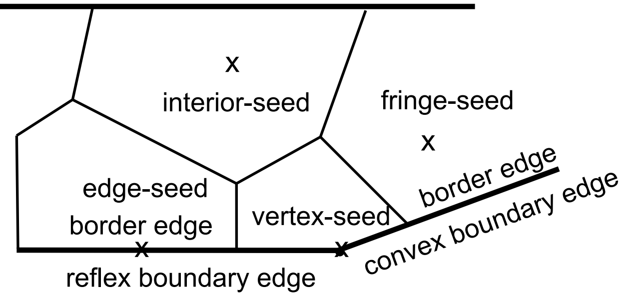
- Get
  - Voronoi mesh of convex polyhedral cells
  - Bounded cell aspect ratio and facet dihedrals
  - Random orientation of mesh edges
    - Needed for fracture mechanics where cracks are restricted to edges





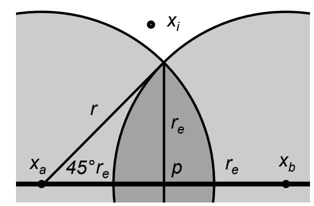
Boundary Sampling

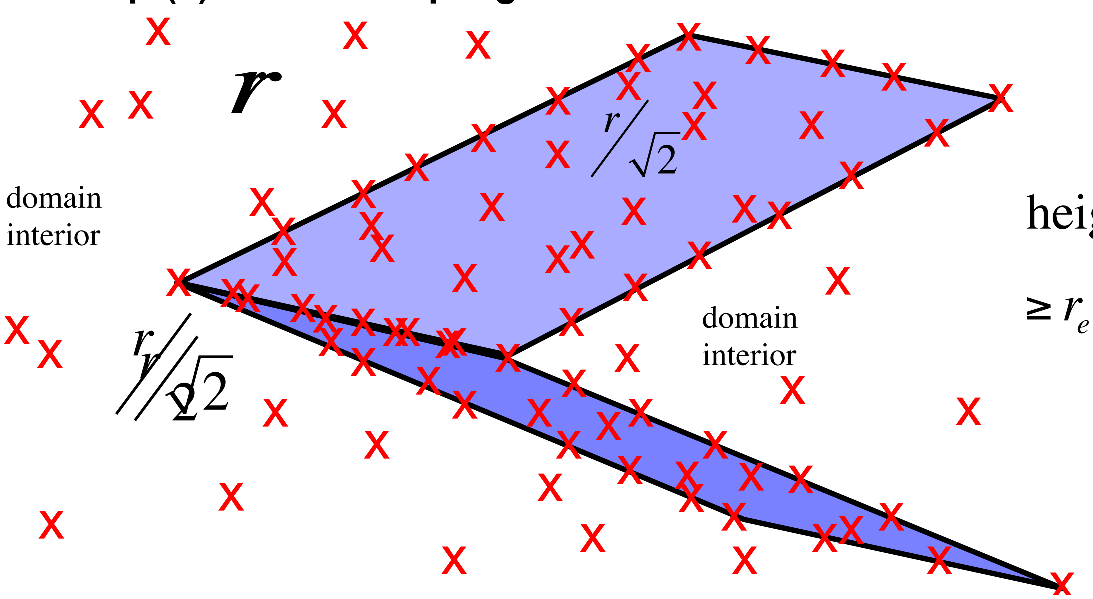
- **Maximally sample**
  - **Points interior to domain, not on boundary…**
    …unless we have to:
    - **Reflex features require special care, not sharp ones**
      - **“Reflex” includes 2-sided facets**
    - **Not the dual of a body-fitted primal mesh**
      - **Better (not constant 90°) dihedrals at boundary**

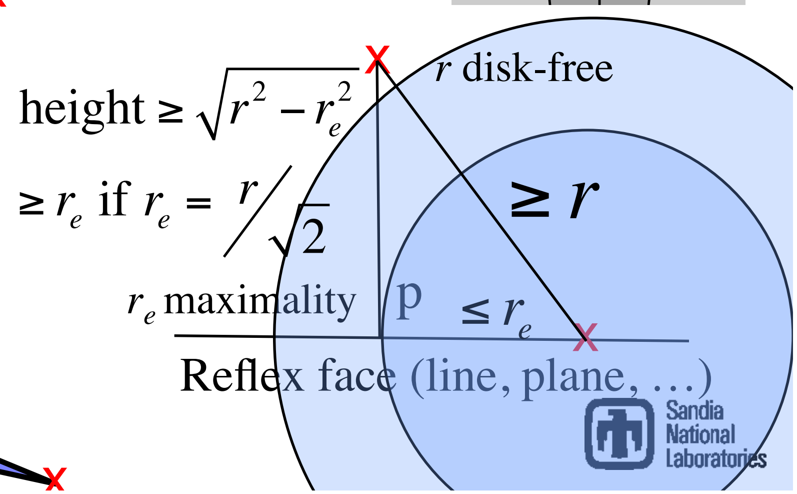


- **Goal: cells align with boundary features, cells are convex**
- **Sufficient: every point on a reflex face is closest to a sample from that reflex feature (or sub-facet)**
  - **Sqrt(2) denser sampling on reflex feature**

$$r_e = r/\sqrt{2}$$

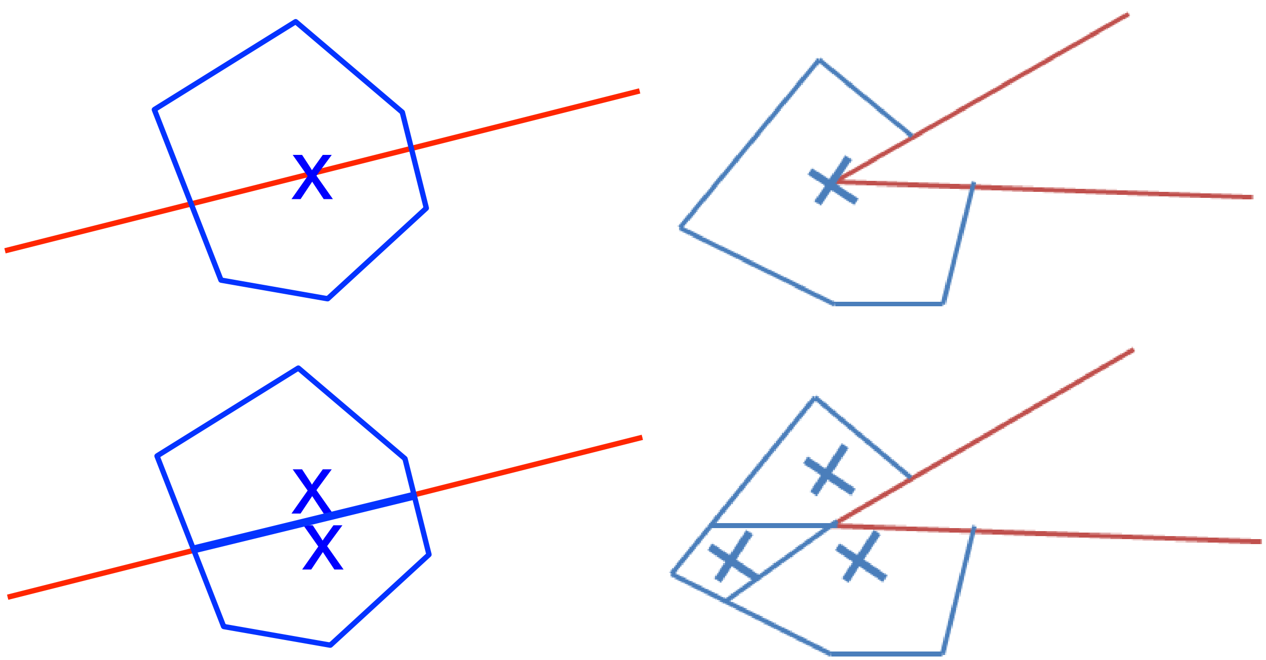


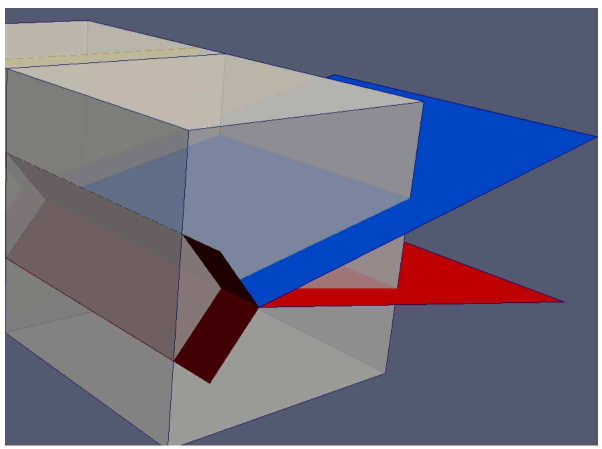




Bonus: Convex Cells
Paper: star-shaped cells at reflex faces

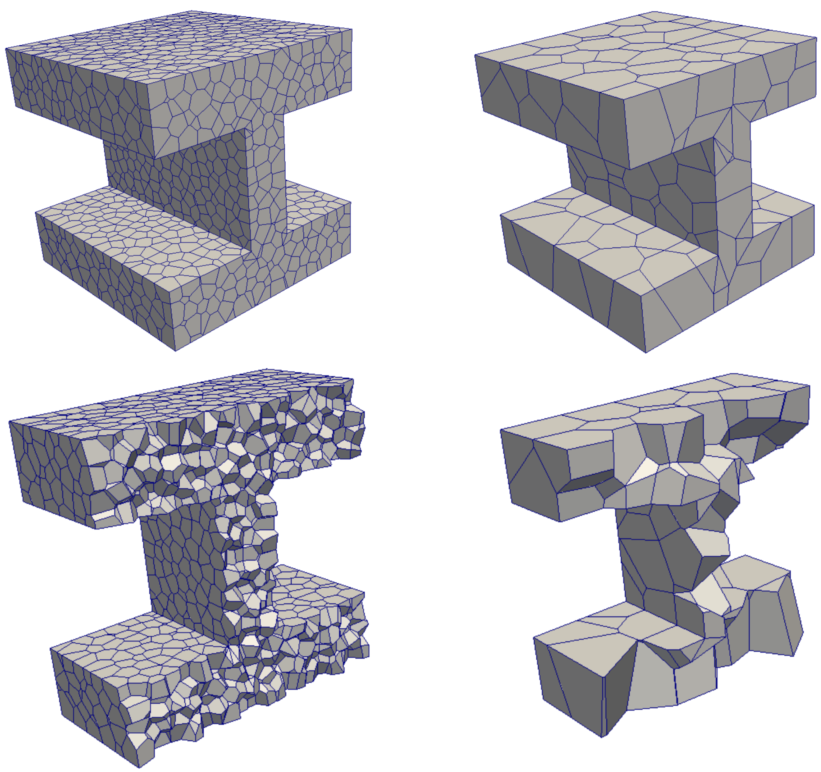
- Clipping by boundary
  - By prior page only non-reflex (convex) boundary features affect interior samples
  - Intersection of convex Voronoi cell w/ convex boundary = convex clipped cell
- Symbolic duplication of reflex samples

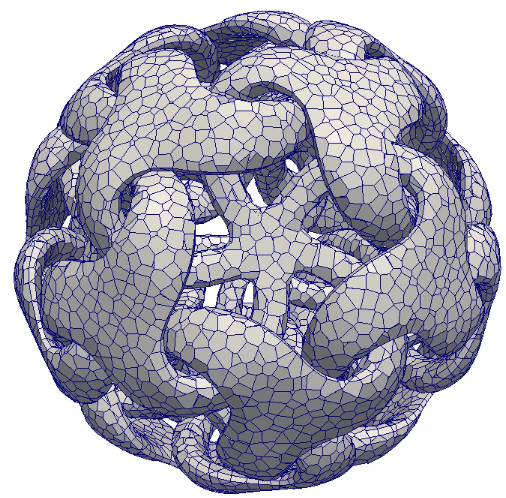




Voronoi Quality

- Provable facet dihedral angle bounds
- Provable cell aspect ratios





Sandia National Laboratories

Quality proof idea

~~Bias free: $\forall \Omega \subset \mathcal{D}_{i-1} : P(x_i \in \Omega) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathcal{D}_{i-1})}$ (1a)~~

Empty disk: $\forall x_i, x_j \in X, i \neq j : ||x_i - x_j|| \geq r$ (1b)

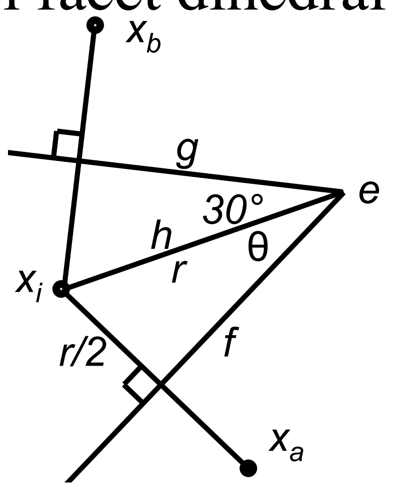
Maximal: $\forall p \in \mathcal{D}, \exists x_i \in X : ||p - x_i|| < r$ (1c)

Voronoi:

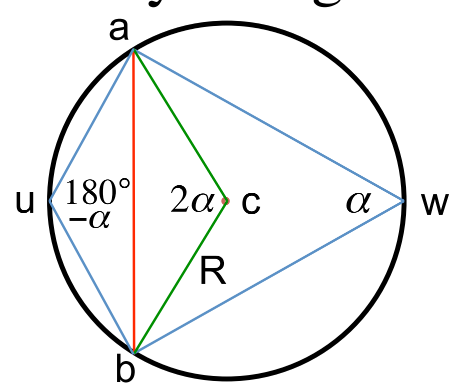
$V_i = \{p\} \in \mathcal{D} : \forall j, ||p - x_i|| \leq ||p - x_j||$

- **"Maximality" bounds the maximum distance from Voronoi cell seed to its vertices**
  = Delaunay vertex to circle center
- **"Disk-free" bounds the minimum distance between two seeds**
  = a Delaunay edge

Voronoi facet dihedral angles:



Delaunay triangle angles:



(b) Central Angle Theorem.

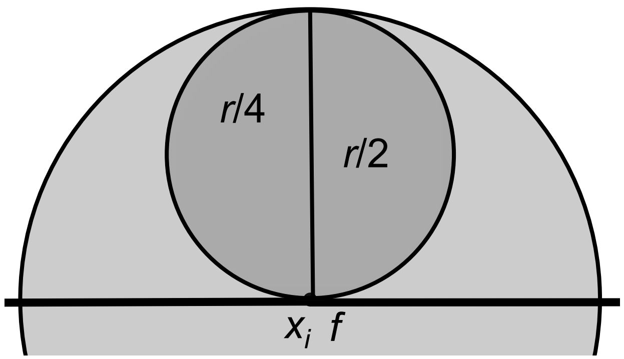
as Chew 89

Sandia National Laboratories

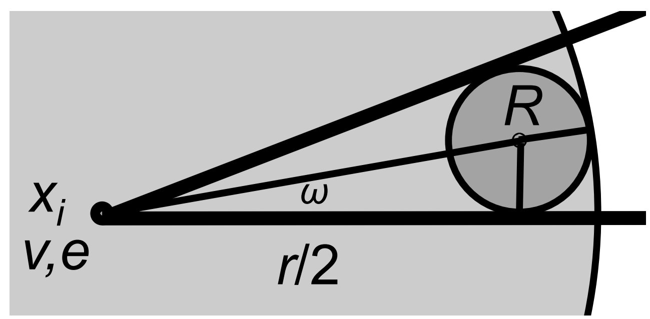
Aspect Ratio Proofs (star-shaped cells)

- **Aspect ratio**
  - **Circumscribed sphere radius < r** **(from maximality)**
  - **Inscribed sphere radius > some factor r** **(from disk-free)**

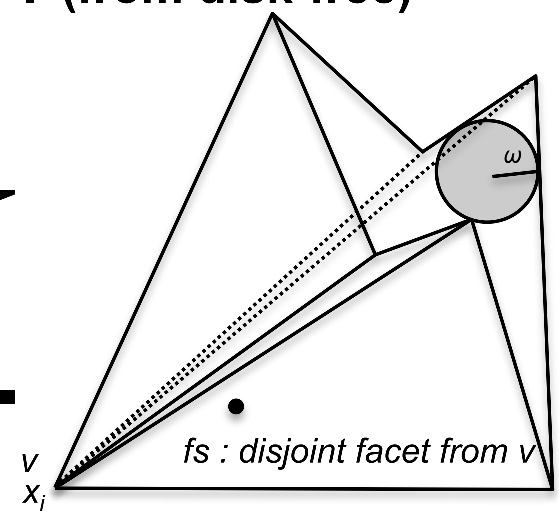
    If cell is interior: r/2



Clipped by one facet: r/4



Facets of one edge

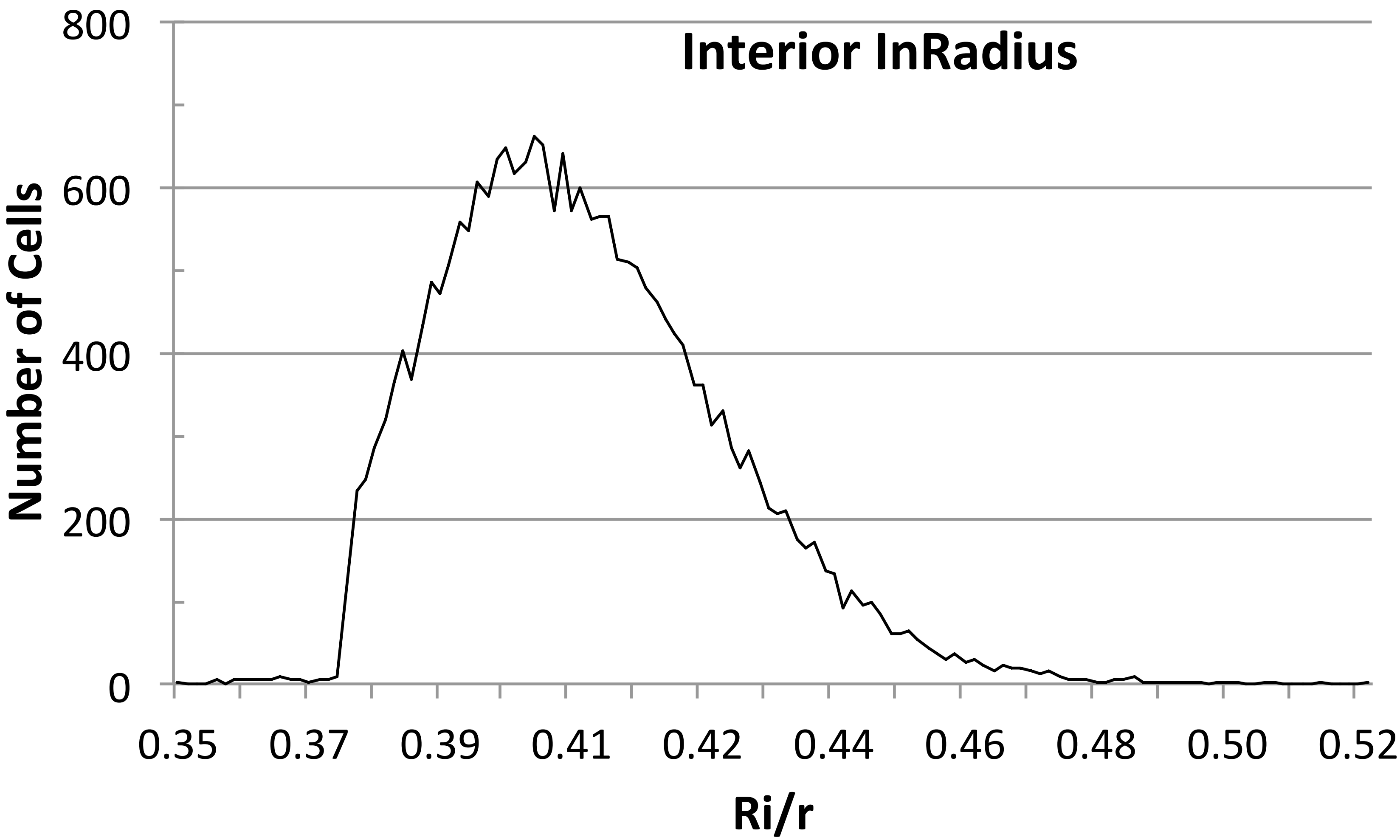


Facets of one vertex

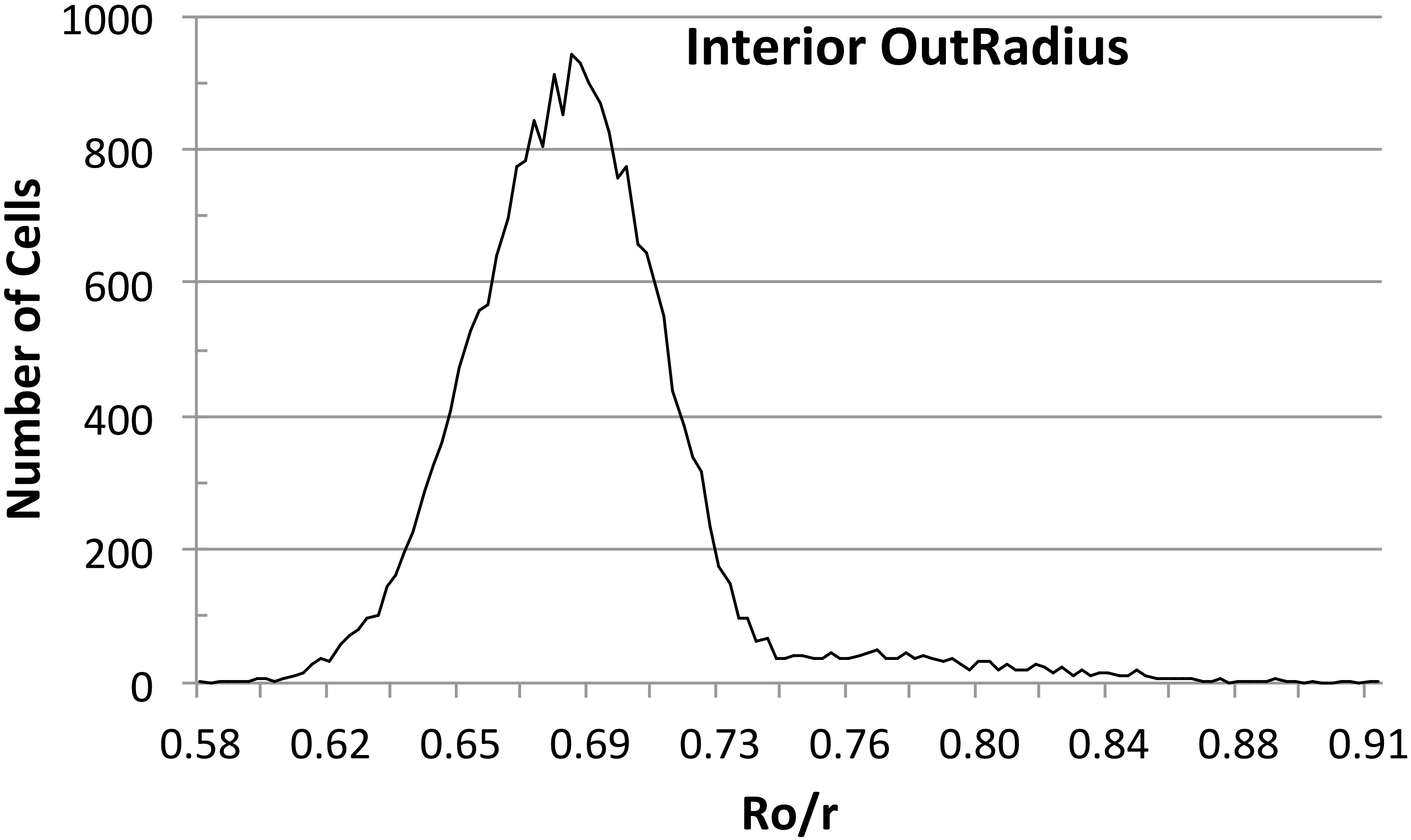
Disjoint facets: feature size *fs*

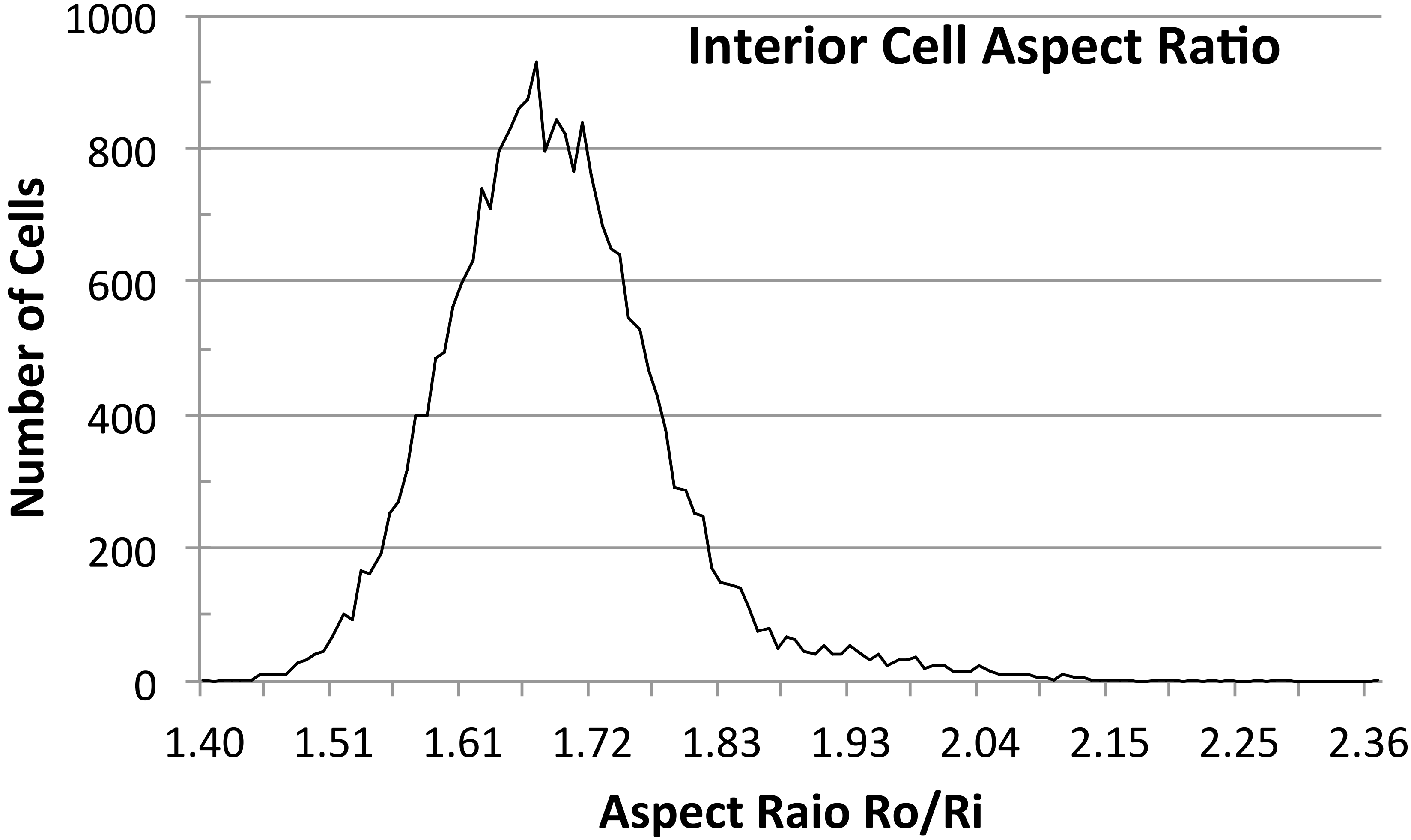
$$A \le 4 \max\left(\sqrt{2}, r / fs\right) \max\left(1, \frac{1+\sin\omega}{2\sin\omega}\right)$$

Interior cells



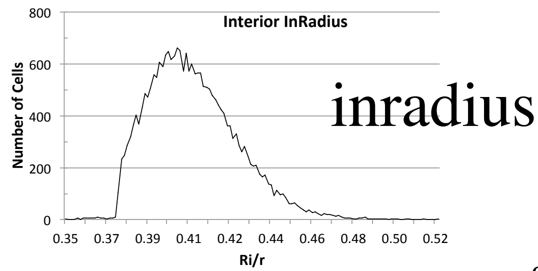
Interior cells

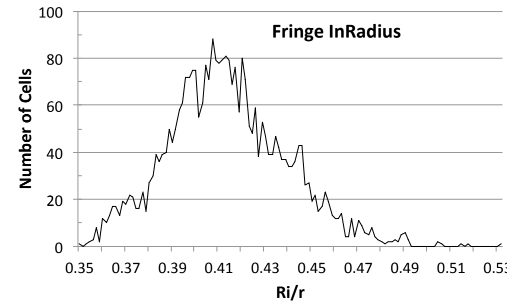



Interior Cell Aspect Ratio
Number of Cells
1000
800
600
400
200
0
1.40
1.51
1.61
1.72
1.83
1.93
2.04
2.15
2.25
2.36
Aspect Raio Ro/Ri

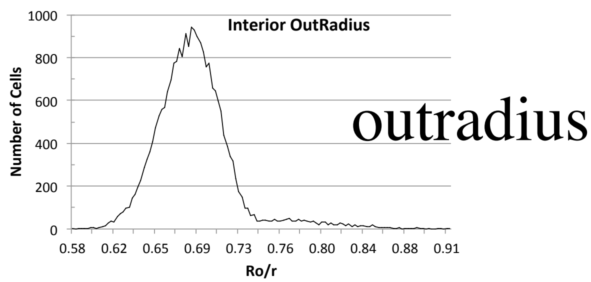
Observed Aspect Ratio

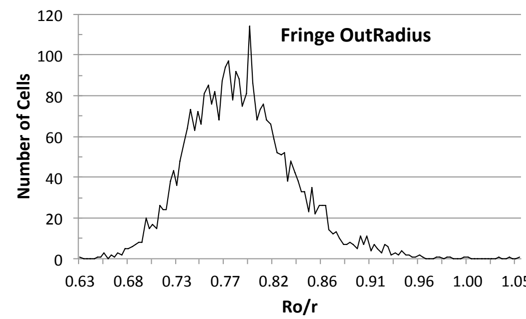
$$A \le 4\max\left(\sqrt{2}, r/fs\right)\max\left(1, \frac{1+\sin\omega}{2\sin\omega}\right)$$

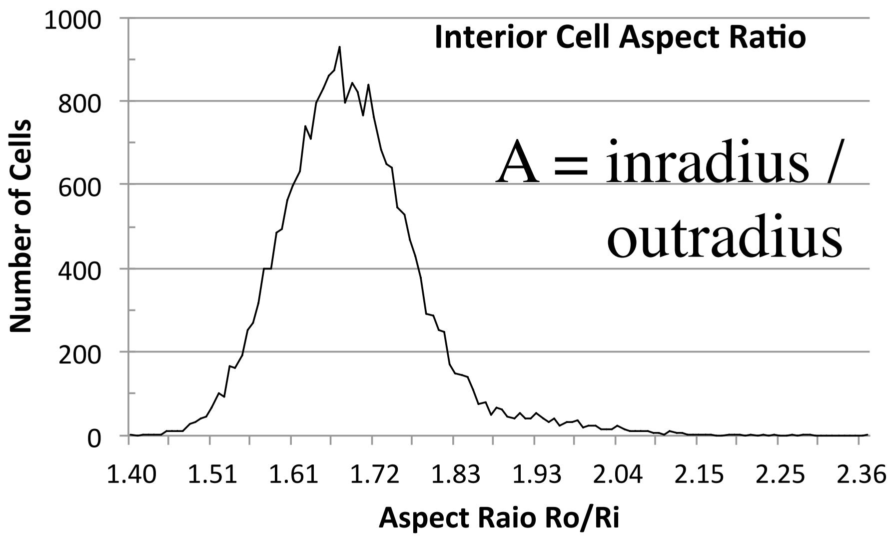


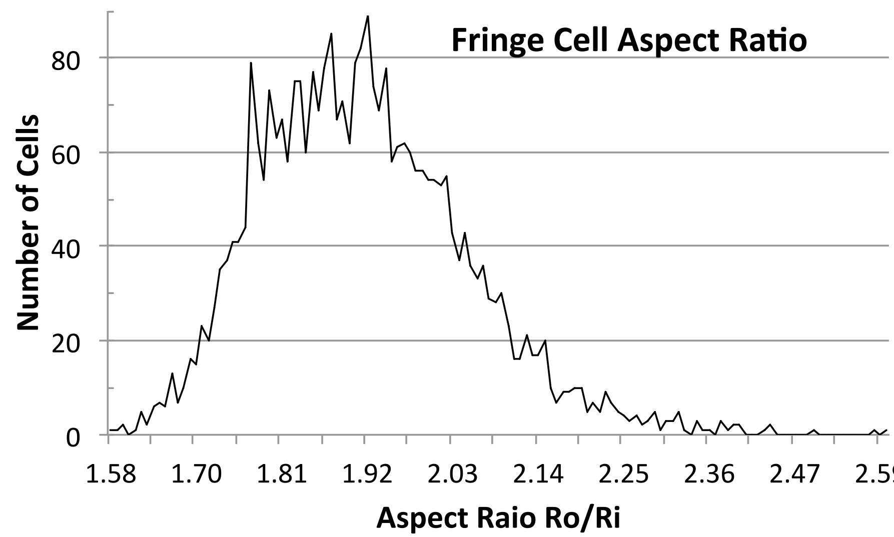


star-shaped cells

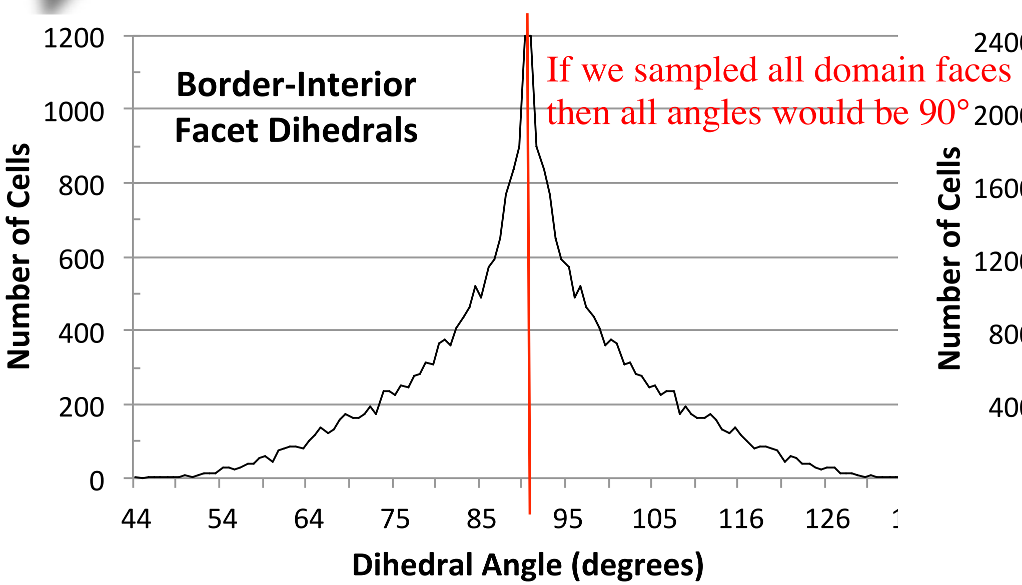


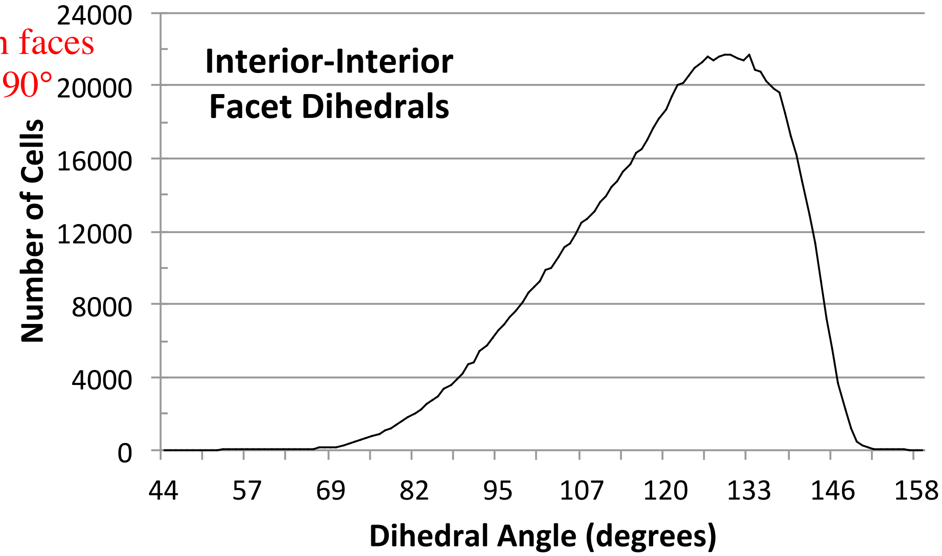






Quality plots
Dihedral Angles



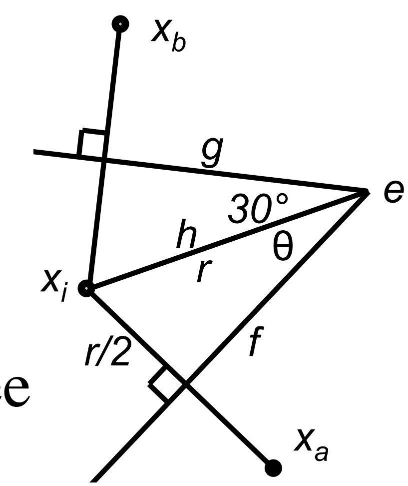


provably $\in [30°, 150°]$ near one border facet

provably $\in [20.7°, 159.7°]$ otherwise

provably $\in [60°, 150°]$

Recall proofs idea:
Distance from seed to
  cell vertex bounded above by maximality
  cell facet distance bounded below by disk-free

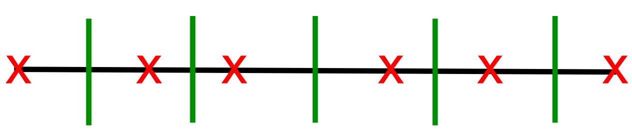


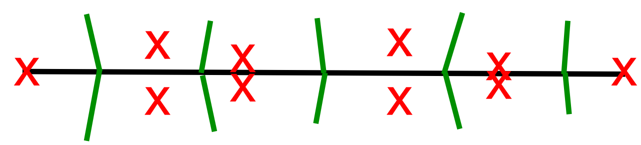
Sandia National Laboratories

Quality: what's missing?

Work in progress:

- **Short edges**
  - **Collapsed, leading to non-planar faces**
  - **OK for Joe Bishop fracture simulation but not ideal**
- **Voronoi *facet* aspect ratio bounds**
  - **Smoothing or sample insertion constraints may fix**
- **90° facet dihedrals between samples *on reflex faces.* (Recall no samples *on* other faces)**
  - **Small random perpendicular offsets may fix**





Conclusions

- w/ Patney, Davidson, Owens (UC Davis)
- w/ Knupp, Bishop, Martinez, Leung (SNL)

- 1. Maximal Poisson-disk sampling point clouds
  - Essence: First provable maximal, bias-free, O(n) space, E(n log n) time
  - Impact: Graphics hot topic (texture synthesis). Ensemble calculations for V&V

  2. Triangular meshes
  - Essence: Provable quality bounds from random points
  - Impact: Seismic simulations

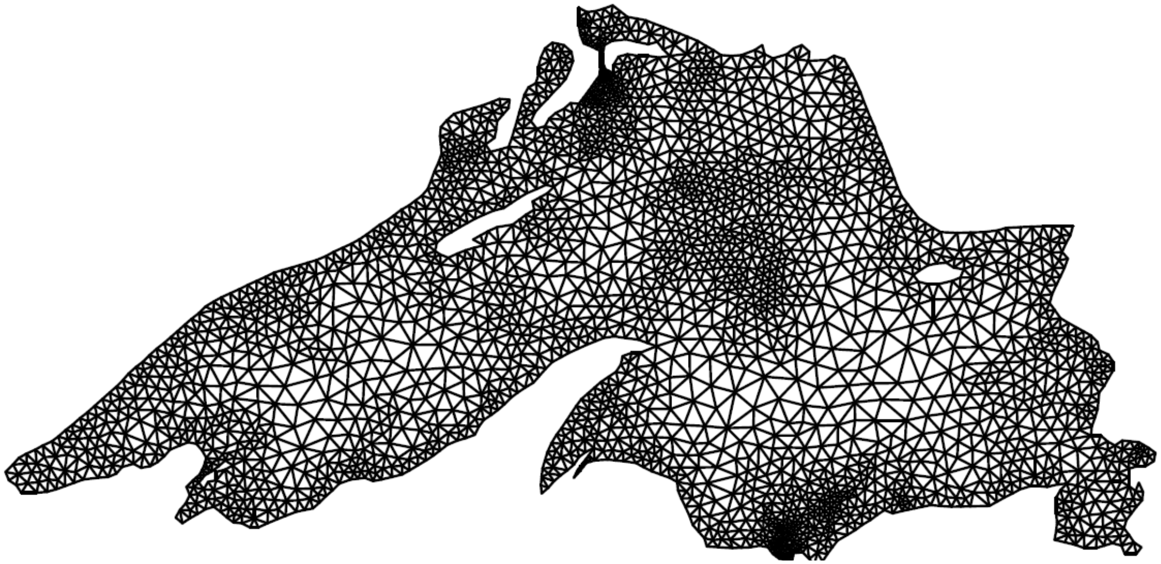
  3. Voronoi meshes
  - Essence: NOT the dual of a boundary-fitted triangulation
  - Impact: Fracture simulations

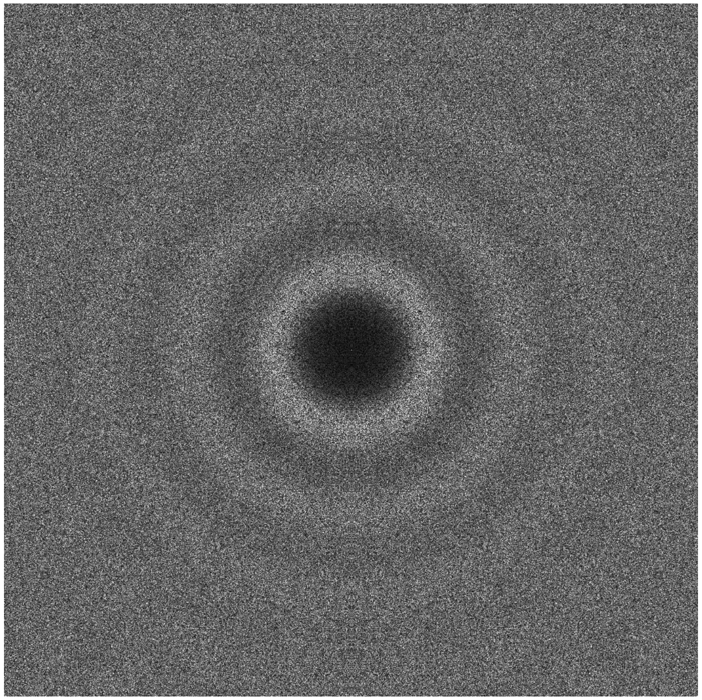
Efficient Maximal Poisson-Disk Sampling.
Ebeida, Patney, Mitchell, Davidson, Knupp & Owens.
SIGGRAPH 2011. ACM Transactions on Graphics.

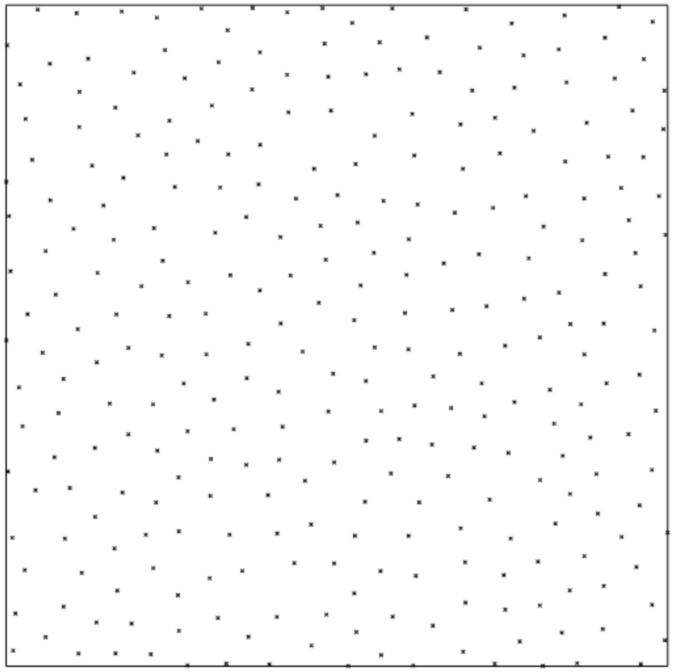
Efficient and Good Delaunay Meshes From Random Points.
Ebeida, Mitchell, Davidson, Patney, Knupp & Owens.
SIAM Conference on Geometric and Physical Modeling.
J Computer-Aided Design special issue.

Uniform Random Voronoi Meshes.
Ebeida & Mitchell.
International Meshing Roundtable, Oct 2011.

- ***Community should consider using maximal samples for mesh points****… even if Poisson-disk process isn't important*
  - ***Better sizing control.***
  - ***Never O(n²)***
    - ***To do: study element count and grading vs. Delaunay refinement.***

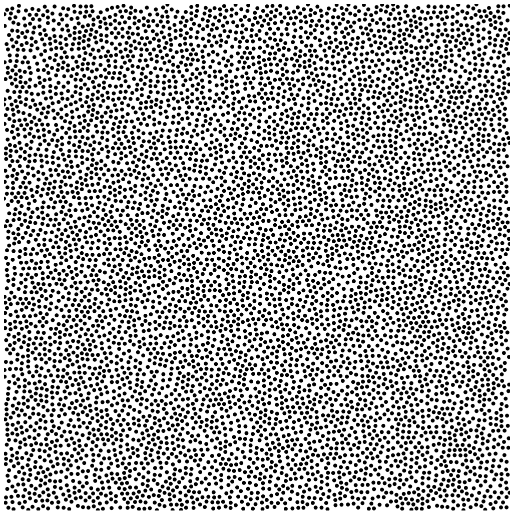
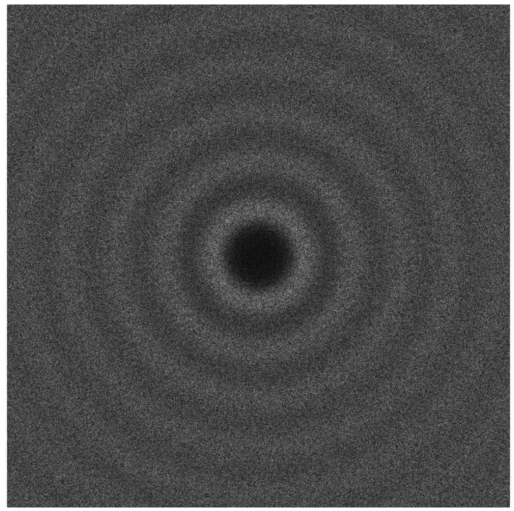
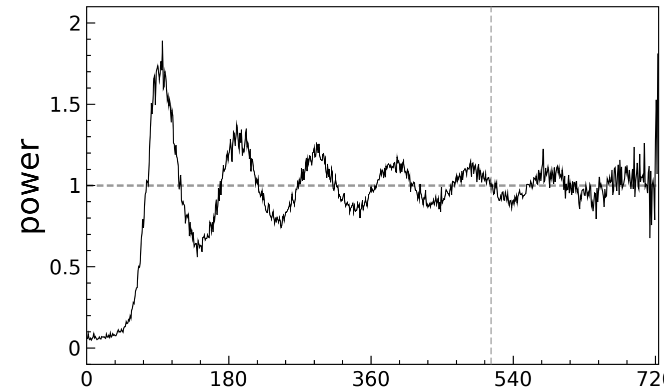




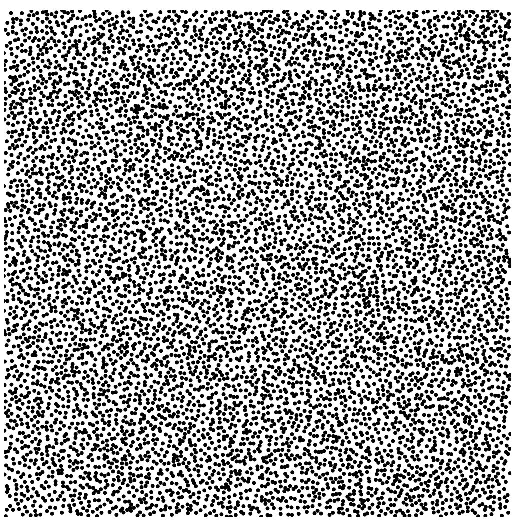
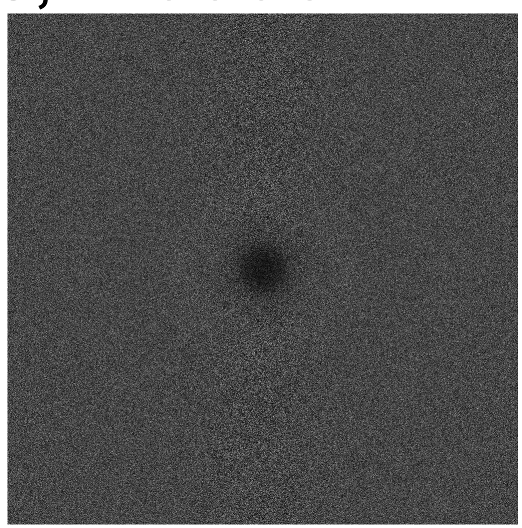
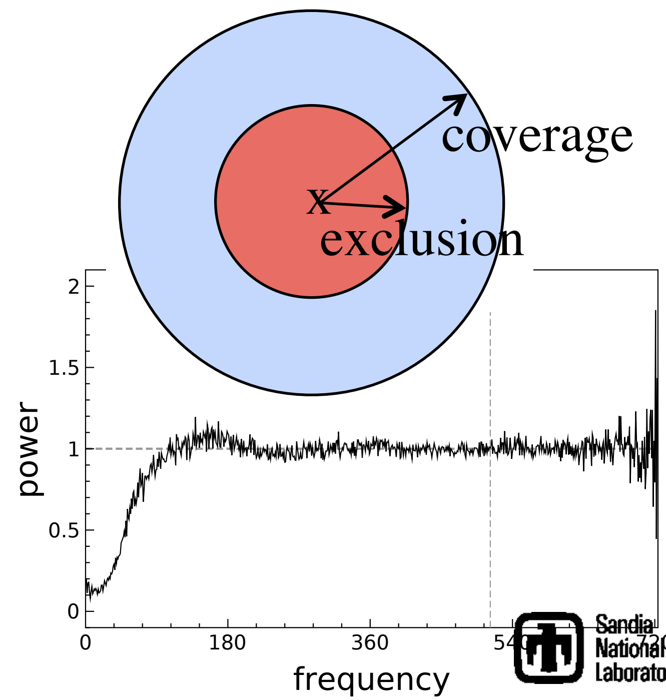


What is the real goal?

- **Classic MPS – a lot of effort to get maximal**

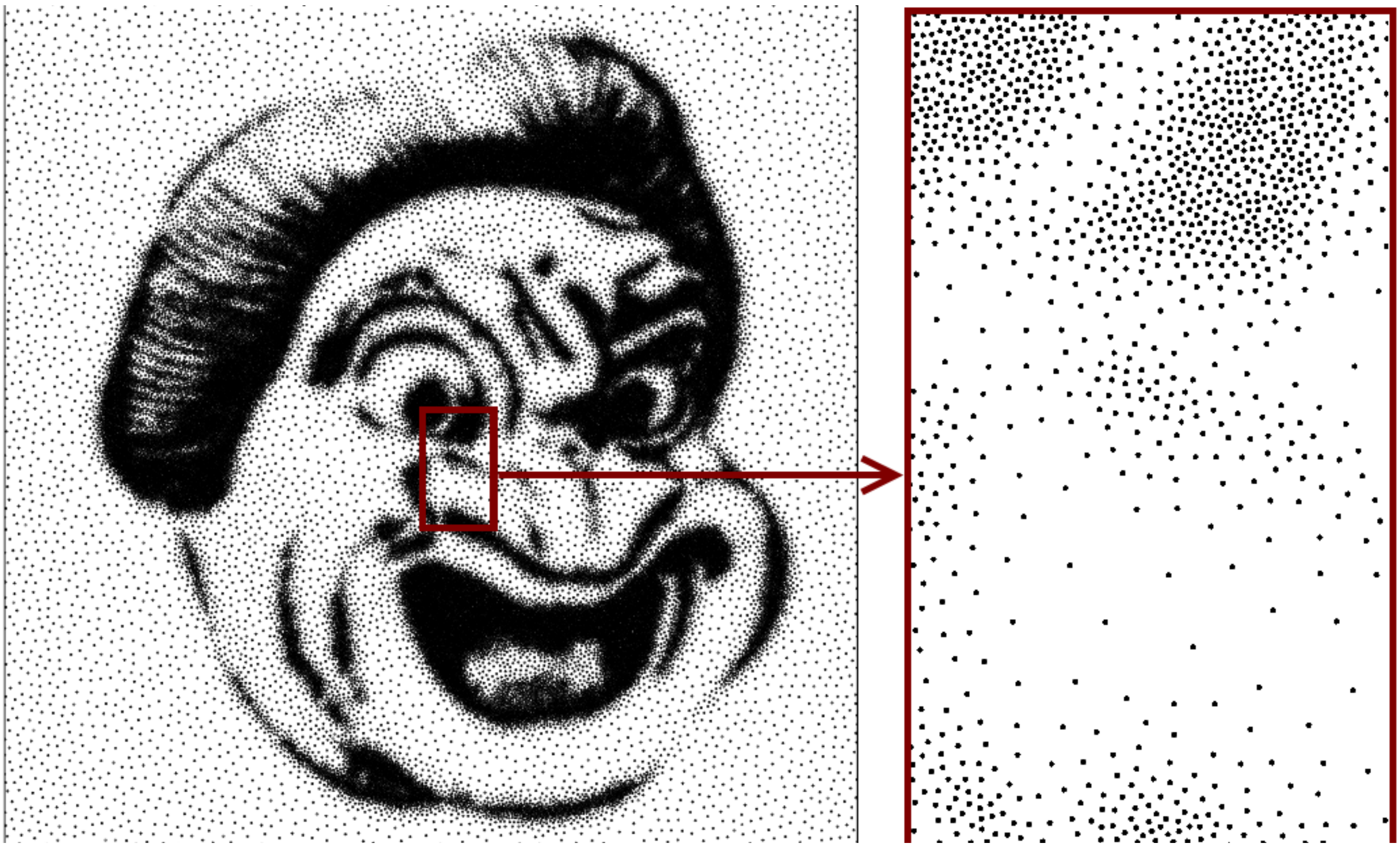




- **Two-radii MPS, in CCCG**

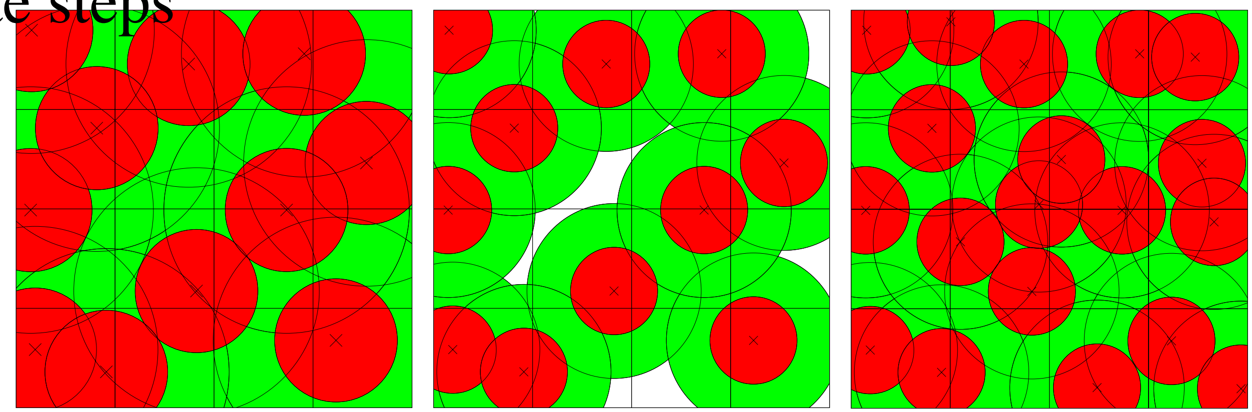
Sandia National Laboratories

Spatially-Variable radius MPS



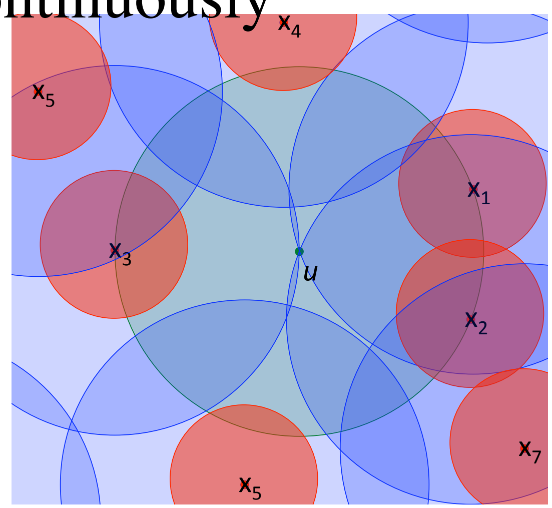
Hierarchical by shrinking radius

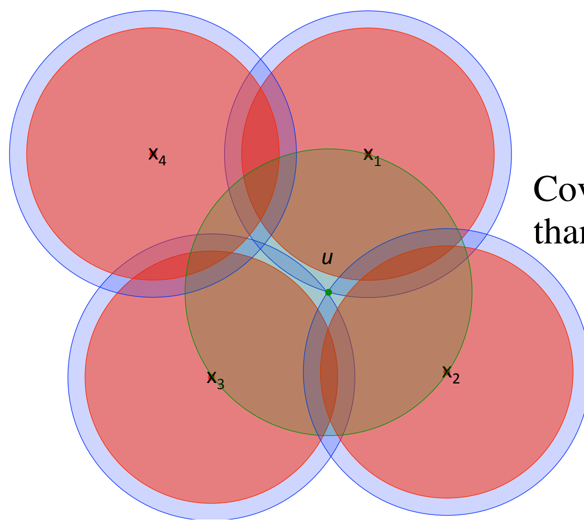
Discrete steps



(a) $t = 0.8$ end (b) $t = 0.6$ start (c) $t = 0.6$ end

Continuously

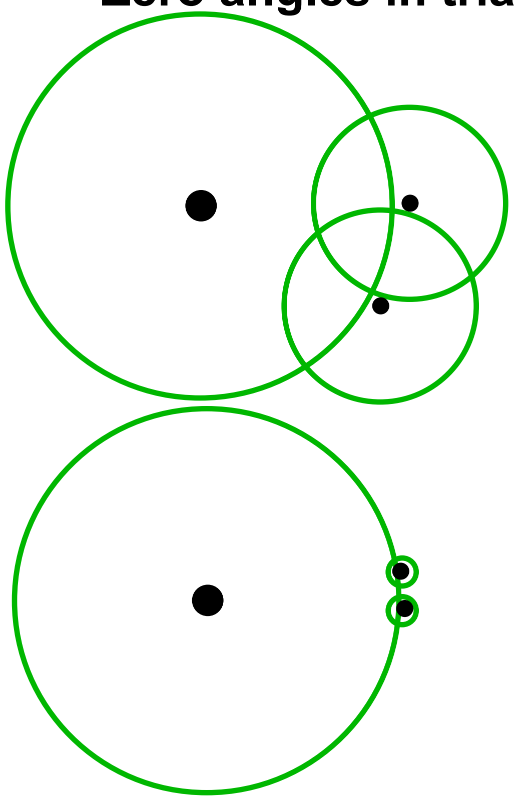


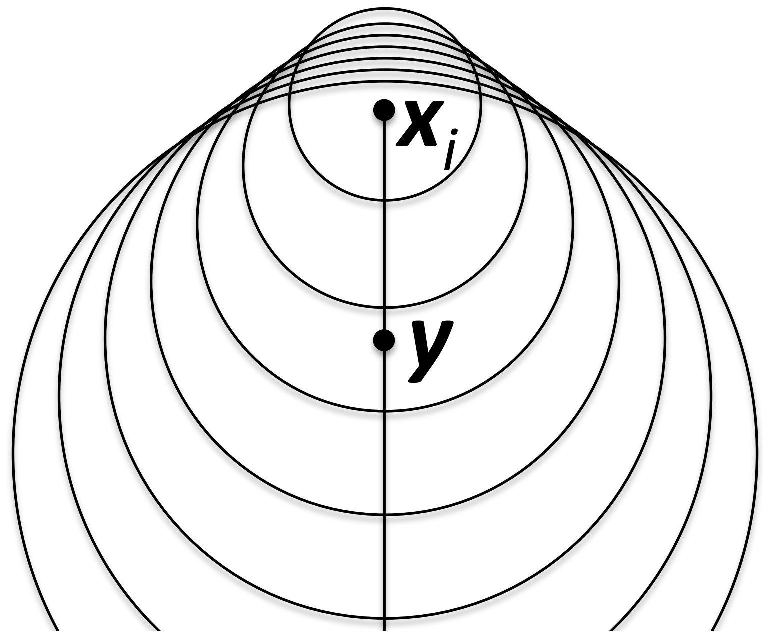


Coverage radius (blue) larger than inhibition radius (red)

How fast can it vary?

- Shrink too fast, number of neighbors is unbounded
  - Infinite run-time
  - Zero angles in triangulation





How fast can it vary?

Method	Distance Function	Order Independent	Full Coverage	Conflict Free	Edge Min	Edge Max	Sin Angle Min	Max L
Prior	$r(\mathbf{x})$	no	no	no	$1/(1+L)$	$2/(1-2L)$	$(1-2L)/2$	$1/2$
Current	$r(\mathbf{y})$	no	no	no	$1/(1+L)$	$2/(1-L)$	$(1-L)/2$	1
Bigger	$\max(r(\mathbf{x}), r(\mathbf{y}))$	yes	no	yes	1	$2/(1-2L)$	$(1-2L)/2$	$1/2$
Smaller	$\min(r(\mathbf{x}), r(\mathbf{y}))$	yes	yes	no	$1/(1+L)$	$2/(1-L)$	$(1-L)/2$	1

Where L is Lipschitz constant: f(x)-f(y) < L |x-y|

Sandia National Laboratories